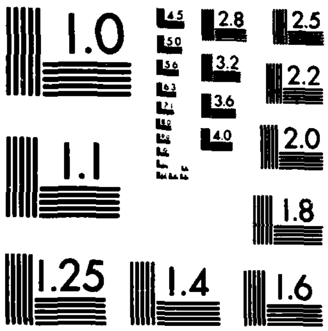
AD-A127 021 FAULT DIAGNOSIS OF NONLINEAR ANALOG CIRCUITS VOLUME V 1/1
HAFDIC: A PROGRAM F..(U) PURDUE UNIV LAFAYETTE IN
SCHOOL OF ELECTRICAL ENGINEERING Y S ELCHERIF ET AL.
UNCLASSIFIED APR 83 N00014-81-K-0323 F/G 9/5 NL




MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Electrical Engineering

# FAULT DIAGNOSIS OF NONLINEAR ANALOG CIRCUITS

## VOLUME V
## HAFDIC: A PROGRAM FOR GENERATING A HARD FAULT DICTIONARY


Y.S. Elcherif
P.M. Lin





School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907


April 1983




This work was supported by Office of Naval Research,
Contract No. N00014-81-K-0323.

# FAULT DIAGNOSIS OF NONLINEAR ANALOG CIRCUITS

## VOLUME V

## HAFDIC: A PROGRAM FOR GENERATING A HARD FAULT DICTIONARY



Y. S. Elcherif
P. M. Lin

April 1983

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

This work was supported by Office of Naval Research,
Contract No. N00014-81-K-0323.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|---|
| 1. REPORT NUMBER | 2. GOVT ACCESSION NO. | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle) Fault Diagnosis of Nonlinear Analog Circuits, Volume V, HAFDIC: A Program for Generating a Hard Fault Dictionary | | 5. TYPE OF REPORT & PERIOD COVERED |
| | | 6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s) Y. S. Elcherif, P. M. Lin | | 8. CONTRACT OR GRANT NUMBER(s) N00014-81-K-0323 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS Purdue University School of Electrical Engineering West Lafayette, IN 47907 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Arlington, Virginia | | 12. REPORT DATE April 1983 |
| | | 13. NUMBER OF PAGES 76 |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) SAME | | 15. SECURITY CLASS. (of this report) Unclassified |
| | | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release. Purdue Research Foundation reserves the copyright, but grants governmental agencies royalty - free right to use the program.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

SAME

18. SUPPLEMENTARY NOTES

NONE

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This volume contains the user's guide and the listing of HAFDIC, a program for generating hard fault dictionary for nonlinear analog cirucits. The theoretical foundation for this program was described in two previous volumes of the report entitled "Fault Diagnosis of Nonlinear Analog Circuits":

Vol. I. DC Diagnosis of Hard Failures, P.M. Lin and Y.S.Elcherif, July 1982.

Vol. IV An Isolation Algorithm for the Analog Fault Dictionary, Y. S. Elcherif and P. M. Lin, April 1983.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

## Table of Contents

## Introduction

Simulation of analog hard faults in piecewise linear networks could be done efficiently using the N-port theory for formulating the network equilibrium equations then using the complementary pivot theory to solve for the node voltages [1]. For a network of n ports, a set of n hybrid equations could be obtained using Lin's method [2]. These equations act as a data base for simulating different faults without the need to reformulate new equilibrium equations. Every time a fault is simulated, only a subset of these equations have to be solved using the Lemke's complementary pivot algorithm [3] to obtain the node voltages of the preselected test nodes. An analog fault dictionary could be obtained by logical means whereby faults are identified by numerical codes [4]. A FORTRAN program is included in this document to achieve the above tasks. The program user is expected to provide a piecewise linear model of the network to be analyzed.

## Hardfault Modelling:

Open circuited inductances and short circuited capacitances are conveniently simulated without extra work, whereas simulating faults in other elements require the addition of switches which are normally open (NO) or normally closed (NC). Nodes which are chosen as test nodes must be considered to form a zero value current source directed from the test node to the ground node. This is for compatibility with the formulation to make it possible to solve for the node voltages by letting the zero value current source be a current port in the n-port formulation. The basic idea is to pull the following elements out of the network to be considered as current or voltage ports as shown in Fig. 1.

1. independent voltage sources.
2. independent current sources.
3. normally open switches.
4. normally closed switches.
5. ideal diodes.
6. zero value current sources.
7. inductances.
8. capacitances.

Fig. 2 shows possible ways of modelling some common faults. It can be seen that insertion of a normally closed switch requires the addition of a new node. Similar ideas can be used in simulating other types of faults. For example, figure 3 suggests a way for simulating an open circuited base or a short circuited base emitter junction in a piecewise linear model of a bipolar transistor.

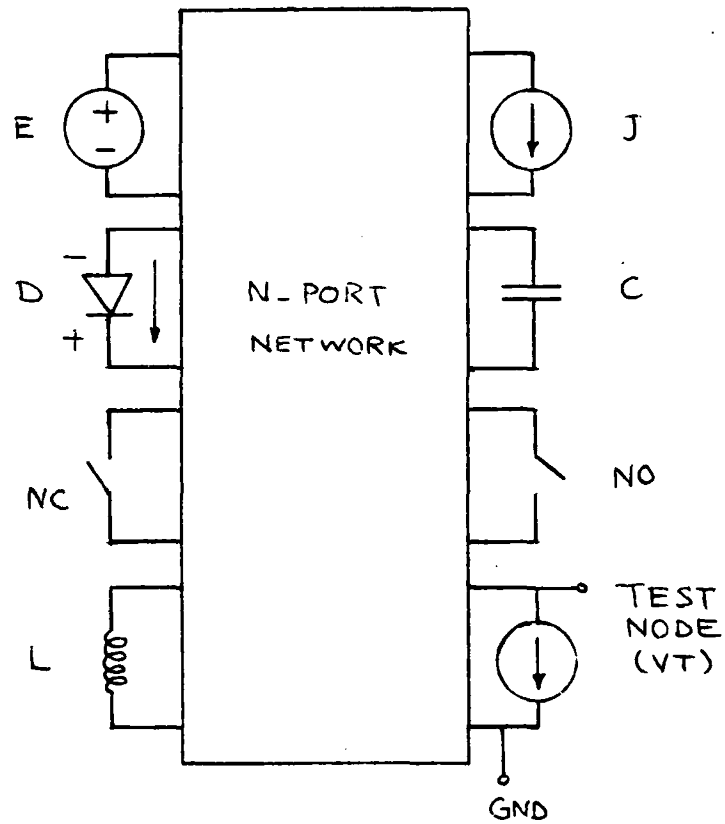



**Fig. 1. Port types in N-port networks.**

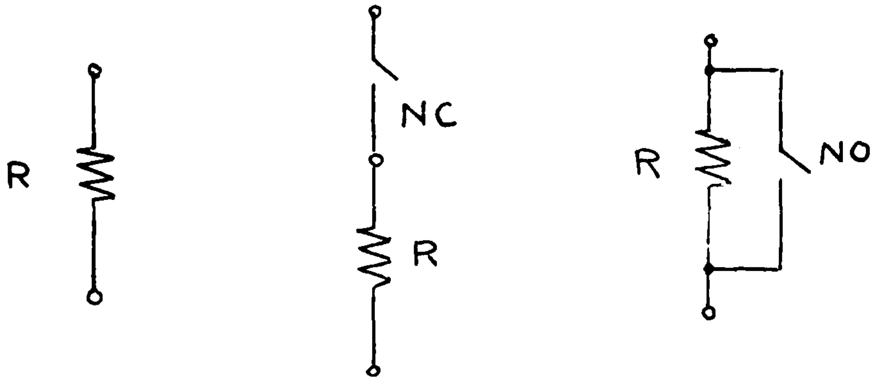



Fig. 2. Simulation of open and short circuits in resistances.

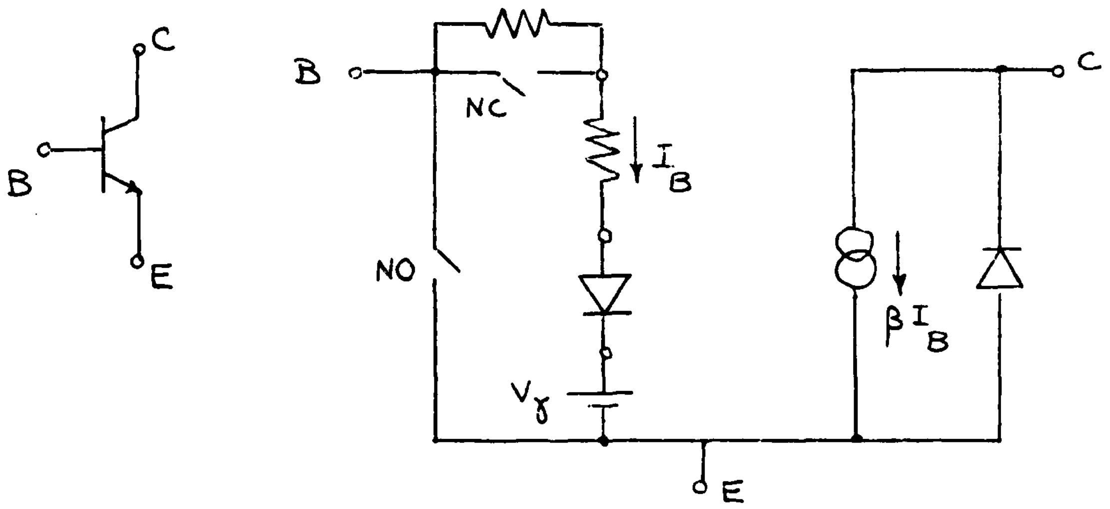



Fig. 3. Simulation of open base and short circuited base emitter junction in piecewise linear models of bipolar transistor.

**Description and Execution of the program:**

The program requires the network description to be stored in a file named data in a special format. A program "cctl" has been written to accept the network description from the user interactively and prepare the required file. Any changes or corrections in the data file afterwards must preserve the required format. The program "cctl" is compatible with the UNIX f77 compiler. To prepare "cctl" from the source file cctl.f, use the following command sequence

```
f77 cctl.f
mv a.out cctl
cctl
```

The current version of the main program is compatible with the MNF compiler that is available on the PVCC CDC 6600 computer system under the control of a dual MACE operating system. Files can be moved back and forth between the ECN computers and the PUCC computer using the UNIX "pjs" command. For complete information about pjs, see the help file available on the UNIX by typing $help pjs. Generally the following forms of the command are enough to put a file in or get a file from the PFILES storage in PUCC:

$pjs - put - use [id] filename1: filename2

or

$pjs - get - use [id] filename1: filename2

where

id: is the user id in the PUCC.

filename1: is the UNIX file name

filename2: is the PFILES file name

If the filename where data is to be transferred is not in the system, a new file will be created and given the specified name. The computer will then prompt by asking about the account number and password on the PUCC computer. The main program HAFDIC uses the following 4 main subroutines that are available in 4 separate files:

1. HYBRID
2. LEMKE
3. AMBSET
4. FTCODE

None of these subroutines requires simultaneous availability of another subroutine in the core which makes possible overlay loading in a smaller core size if so desired. Without overlay loading, the required core size is 134000 words.

The simplest way of execution given the source files is to compile every file separately and keep a copy of the binary object files resulting from compilation to be loaded individually any time a program is needed.

The deck required for compiling HYBRID is:

```
12345, ABC, MF77000, CM77000.
PFILES ,, HYBRID.
RFL (77000)
MNF (I = HYBRID, B = BHYBRD, N, L = 0
PFILES, PUT, BHYBRD.
#EOR
#EOF
```

The parameter N in the MNF command will prevent execution and L = 0 will suppress listing. The maximum field length and central memory size required for compiling any subroutine are those required for HYBRID. Therefore the same deck can be used for compiling other subroutines. The names of the binary files available now on the system are BHAFDIC, BHYBRD, BLEMKE, BAMBST, BFTCOD. It has to be noted that the commands PFILES and MNF automatically set the field length to the default values of 15000 and 45000 respectively. The 45000 words field length is not enough for compiling some subroutines. Therefore it is necessary to use the RFL command for adjusting the field length. The deck required for that is:

```
ABC, 12345, MF60000, CM60000.
PFILES,, AMBSET.
RFL,60000.
MNF(I = AMBSET, B = BAMBST, N,L = 0)
PFILES, PUT, BAMBST.
PFILES ,, FTCODE.
RFL, 60000.
MNF(I = FTCODE, B = BFTCOD, N, L = 0)
PFILES, PUT, BFTCOD.
PFILES,, LEMKE.
RFL, 60000.
MNF(I = LEMKE, B = BLEMKE, N,L = 0)
PFILES, PJT, BLEMKE.
PFILES,, TESTN.
RFL, 60000.
MNF(I = TESTN, B = BTESTN, N,L = 0)
PFILES, PUT, BTESTN.
#EOR
#EOF
```

The deck required for execution is

```
12345, ABC, MF134000, CM134000.
PFILES ,, BTESTN.
PFILES ,, BHYBRD.
PFILES ,, BLEMKE.
PFILES ,, BAMBST.
PFILES ,, BFTCOD.
RFL, 134000.
LOAD, BHYBRD.
LOAD, BLEMKE.
LOAD, BAMBST.
LOAD, BFTCOD.
LOADX, BTESTN.
REWIND, OUTPUT
PFILES (PUT, RESULT, X = OUTPUT)
#EOR
#EOF
```

The last two commands will store the output in a file called RESULT which can be printed on the line printer afterwards at the user's convenience, or kept for inspection in the PFILES storage.

**Example**

The video amplifier in Fig. 4 has the piecewise linear model shown in Fig. 5, where branch numbers are enclosed in circles while node numbers are written beside the corresponding nodes. The user-computer dialogue as controlled by "cctl" is shown next. Next to it is the data file produced. Note the second line in the file which contains 14 integer numbers and a floating point number. This line gives the numbers of branches in the 14 permissible types of two terminal elements followed by the ambiguity voltage range. If the user decides to add or delete any branches before running the program, he has to modify the number of branches accordingly. The numbers in the second line follow the same sequence which appears in the question "branch type?(e,L,..etc.)". The program output follows the listing of the data file.

**Simulating Multiple Faults**

If several branches are to be considered simultaneously faulty, they should be all assigned the same fault number as fault 9 in the example. If the fault in some branch is not required to be simulated at all, the field corresponding to the fault number should be skipped when replying to the question concerning this branch, as in branch

48 in the example.

**Branch Polarities:**

The end nodes of all branches, except diodes and controlled branches, can be entered in either order. The values of voltage and current will be adjusted accordingly to be either positive or negative.

For diodes, the node connected to the n terminal should be entered first to make both current and voltage positive in either case of being on or off, otherwise he complementary problem will not be feasible. The program assumes the port voltage and current convention as shown beside the diode element in figure 1.

For control branches, the order of entering the end nodes must be consistent with the order of entering the end nodes of the controlled source according to the current and voltage conventional directions shown in figure 1.

**Numbering Nodes and Branches:**

Branches can be given any numbers which are not necessarily consecutive. However node numbers must be consecutive starting from zero, which is to be the ground node. Branch names can be up to 4 characters which must start by one of the 14 acronyms (e,j,...) shown in the example. The names of the diodes and the test node parts are the ones to be used in the program output.

**References:**

1. P. M. Lin, "DC Fault Diagnosis Using Complementary Pivot Theory," Proc. 1982 IEEE Int'l Symp on Circuits and Systems, pp. 1132-1135, May 1982.
2. L. Chua and P. M. Lin, "Computer-Aided Analysis of Electronic Circuits: Algorithms and Computational Techniques," Englewood Cliffs, NJ: Prentice Hall, 1975.
3. P. M. Lin and Y. S. Elcherif, "Fault Diagnosis of Nonlinear Analog Circuits, Vol. I. DC Diagnosis of Hard Failures," TR-EE 82-21, School of Electrical Engineering, Purdue University, West Lafayette, Indiana, July 1982.
4. Y. S. Elcherif and P. M. Lin, "Fault Diagnosis of Nonlinear Analog Circuits, Vol. IV: An Isolation Algorithm for the Analog Fault Dictionary," Final report to Office of Naval Research, April 1983.

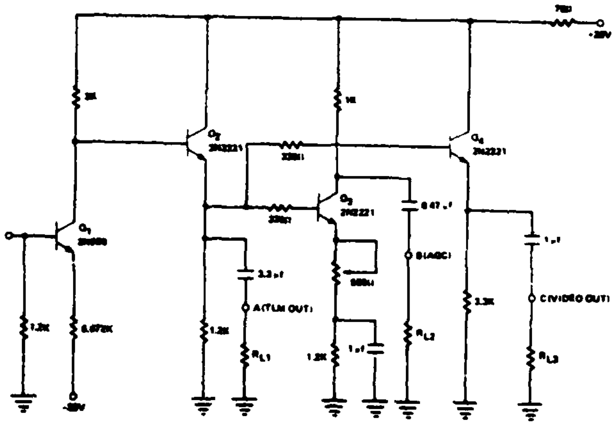



Fig 4 The Video Amplifier

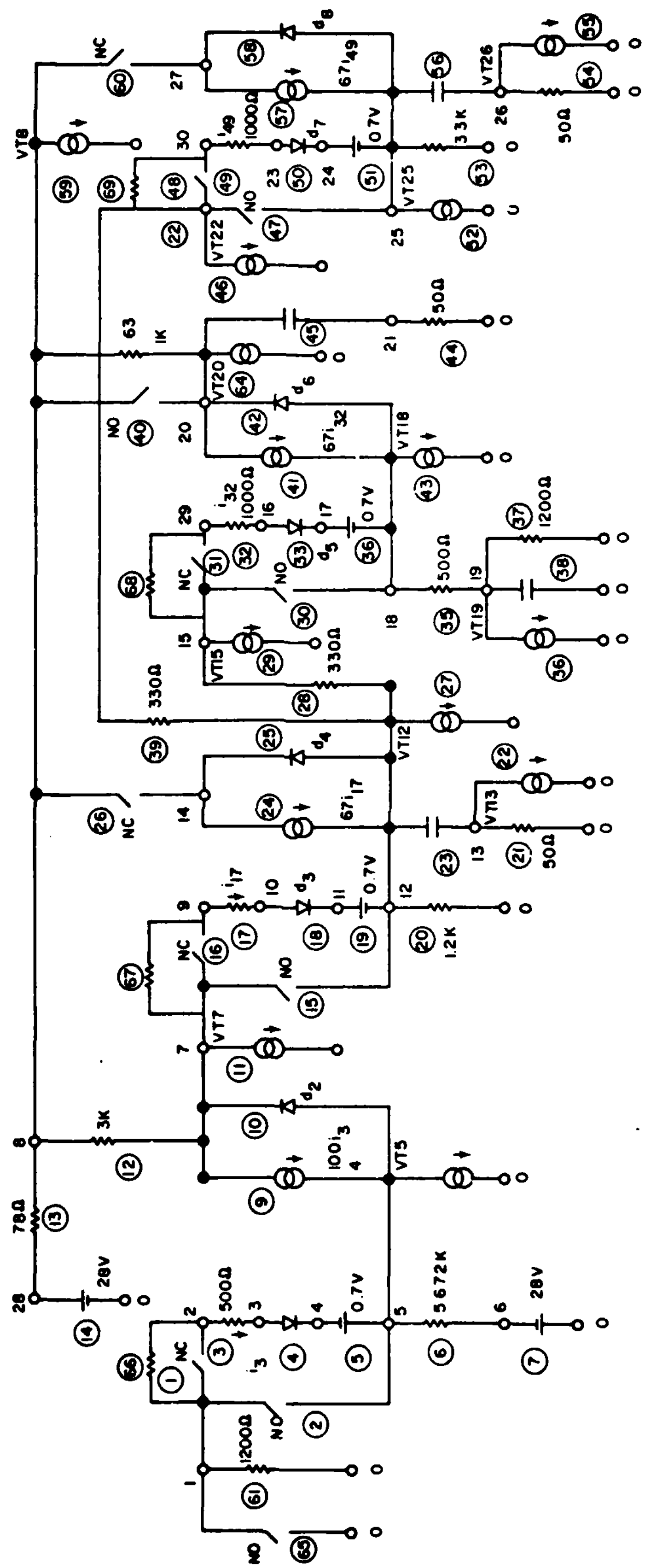


Fig . The piecewise linear model of the video amplifier

```
$ cct1
 problem title:
a new video amplifier circuit
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
e
 branch no,from,to,battery value
7,6,0,-28
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
e
 branch no,from,to,battery value
14,28,0,28
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
e
 branch no,from,to,battery value
5 4 5 .7
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
19,11,12,.7
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
e
 branch no,from,to,battery value
34,17,18,.7
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
e
 branch no,from,to,battery value
51,24,25,.7
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
d1
 branch no,from,to
4,4,3
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
d2
 branch no,from,to
10,7,5
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
d3
 branch no,from,to
18,11,10
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
d4
 branch no,from,to
25,14,12
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
d5
 branch no,from,to
33,17,16
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
d6
 branch no,from,to
42,20,18
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
d7
 branch no,from,to
50,24,23
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
d8
 branch no,from,to
58,27,25
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
nc
 branch no,from,to,fault no(if o.c is to be simulated)
```

```
1,1,2,2
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
nc
 branch no,from,to,fault no(if o.c is to be simulated)
16,7,9,3
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
nc
 branch no,from,to,fault no(if o.c is to be simulated)
26,14,8,6
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
nc
 branch no,from,to,fault no(if o.c is to be simulated)
31,15,29,4
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
nc
 branch no,from,to,fault no(if o.c is to be simulated)
48,22,30,,
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
nc
 branch no,from,to,fault no(if o.c is to be simulated)
60,27,8,7
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
r
 branch no,from,to,resistance value
61,1,0,1200
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
r
 branch no,from,to,resistance value
3,2,3,500
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
r
 branch no,from,to,resistance value
6,5,6,5670
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
r
 branch no,from,to,resistance value
12,7,8,3000
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
r
 branch no,from,to,resistance value
17,9,10,1000
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
r
 branch no,from,to,resistance value
20,12,0,1200
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
r
 branch no,from,to,resistance value
21,13,0,50
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
r
 branch no,from,to,resistance value
13,28,8,78
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
r
 branch no,from,to,resistance value
28,12,15,330
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
r
 branch no,from,to,resistance value
```

```
39,12,22,330
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
r
 branch no,from,to,resistance value
32,29,16,1000
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
r
 branch no,from,to,resistance value
35,18,19,500
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
r
 branch no,from,to,resistance value
37,19,0,1200
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
r
 branch no,from,to,resistance value
40,20,8,1000
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
r
 branch no,from,to,resistance value
44,21,0,50
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
r
 branch no,from,to,resistance value
49,30,23,1000
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
r
 branch no,from,to,resistance value
53,25,0,3300
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
r
 branch no,from,to,resistance value
54,26,0,50
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
r
 branch no,from,to,resistance value
66,1,2,1.e+13
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
r
 branch no,from,to,resistance value
67,7,9,1.e+13
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
r
 branch no,from,to,resistance value
68,15,29,1.e+13
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
r
 branch no,from,to,resistance value
69,23,24,1.e+10
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
cc
 branch no,from,to,control branch,cc value
9,7,5,3,400
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
cc
 branch no,from,to,control branch,cc value
24,14,12,17,67
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
cc
 branch no,from,to,control branch,cc value
```

```
41,20,18,32,67
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
cc
 branch no,from,to,control branch,cc value
57,27,25,49,67
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
vt5
 branch no,from,to
8,5,0
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
vt7
 branch no,from,to
11,7,0
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
vt8
 branch no,from,to
59,8,0
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
vt12
 branch no,from,to
27,12,0
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
vt13
 branch no,from,to
22,13,0
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
vt15
 branch no,from,to
29,15,0
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
vt18
 branch no,from,to
43,18,0
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
vt19
 branch no,from,to
36,19,0
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
vt20
 branch no,from,to
64,20,0
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
vt22
 branch no,from,to
46,22,0
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
vt25
 branch no,from,to
52,25,0
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
vt26
 branch no,from,to
55,26,0
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
c
 branch no,from,to,fault no(if s.c is to be simulated)
23,12,13,13
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
no
 branch no,from,to,fault no(if s.c is to be simulated)
```

```
2,1,5,8
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
no
 branch no,from,to,fault no(if s.c is to be simulated)
65,1,0,9
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
no
 branch no,from,to,fault no(if s.c is to be simulated)
15,7,12,10
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
no
 branch no,from,to,fault no(if s.c is to be simulated)
30,15,18,11
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
no
 branch no,from,to,fault no(if s.c is to be simulated)
63,8,20,9
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
no
 branch no,from,to,fault no(if s.c is to be simulated)
47,22,25,12
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
c
 branch no,from,to,fault no(if s.c is to be simulated)
38,19,0,14
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
c
 branch no,from,to,fault no(if s.c is to be simulated)
45,20,21
,15
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
c
 branch no,from,to,fault no(if s.c is to be simulated)
56,25,26,16
 branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):
stop
ambig set range :
1.4
 file name where data is to be stored:
data
$
```

the first two lines are for program use, DO NOT delete them
5 0 6 8 22 0 4 0 0 0 12 6 4 0 .140e+00
a new video amplifier circuit

| branch type | no | from node | to node | control branch | value | fault no. | for program use only |
|---|---|---|---|---|---|---|---|
| e | 7 | 6 | 0 | | -.280e+02 | | e |
| e | 14 | 28 | 0 | | .280e+02 | | e |
| e | 5 | 4 | 5 | | .700e+00 | | e |
| e | 34 | 17 | 18 | | .700e+00 | | e |
| e | 51 | 24 | 25 | | .700e+00 | | e |
| nc | 1 | 1 | 2 | | | 0 | e |
| nc | 16 | 7 | 9 | | | 0 | e |
| nc | 26 | 14 | 8 | | | 0 | e |
| nc | 31 | 15 | 29 | | | 0 | e |
| nc | 48 | 22 | 30 | | | 0 | e |
| nc | 60 | 27 | 8 | | | 0 | e |
| d1 | 4 | 4 | 3 | | | | e |
| d2 | 10 | 7 | 5 | | | | e |
| d3 | 18 | 11 | 10 | | | | e |
| d4 | 25 | 14 | 12 | | | | e |
| d5 | 33 | 17 | 16 | | | | e |
| d6 | 42 | 20 | 18 | | | | e |
| d7 | 50 | 24 | 23 | | | | e |
| d8 | 58 | 27 | 25 | | | | e |
| r | 61 | 1 | 0 | | .120e+04 | | r |
| r | 3 | 2 | 3 | | .500e+03 | | r |
| r | 6 | 565670 | | | .567e+04 | | r |
| r | 12 | 7 | 8 | | .300e+04 | | r |
| r | 17 | 9 | 10 | | .100e+04 | | r |
| r | 20 | 12 | 0 | | .120e+04 | | r |
| r | 21 | 13 | 0 | | .500e+02 | | r |
| r | 13 | 28 | 8 | | .780e+02 | | r |
| r | 28 | 12 | 15 | | .330e+03 | | r |
| r | 39 | 12 | 22 | | .330e+03 | | r |
| r | 32 | 29 | 16 | | .100e+04 | | r |
| r | 35 | 18 | 19 | | .500e+03 | | r |
| r | 37 | 19 | 0 | | .120e+04 | | r |
| r | 40 | 20 | 8 | | .100e+04 | | r |
| r | 44 | 21 | 0 | | .500e+02 | | r |
| r | 49 | 30 | 23 | | .100e+04 | | r |
| r | 53 | 25 | 0 | | .330e+04 | | r |
| r | 54 | 26 | 0 | | .500e+02 | | r |
| r | 66 | 1 | 2 | | .100e+14 | | r |
| r | 67 | 7 | 9 | | .100e+14 | | r |
| r | 68 | 15 | 29 | | .100e+14 | | r |
| r | 69 | 23 | 24 | | .100e+11 | | r |
| cc | 9 | 7 | 5 | 3 | .400e+03 | | cc |
| cc | 24 | 14 | 12 | 17 | .670e+02 | | cc |
| cc | 41 | 20 | 18 | 32 | .670e+02 | | cc |
| cc | 57 | 27 | 25 | 49 | .670e+02 | | cc |
| vt5 | 8 | 5 | 0 | | | | i |
| vt7 | 11 | 7 | 0 | | | | i |
| vt8 | 59 | 8 | 0 | | | | i |
| vt12 | 27 | 12 | 0 | | | | i |
| vt13 | 22 | 13 | 0 | | | | i |
| vt15 | 29 | 15 | 0 | | | | i |
| vt18 | 43 | 18 | 0 | | | | i |
| vt19 | 36 | 19 | 0 | | | | i |
| vt20 | 64 | 20 | 0 | | | | i |
| vt22 | 46 | 22 | 0 | | | | i |

| | | | | | |
|---|---|---|---|---|---|
| vt25 | 52 | 25 | 0 | | i |
| vt26 | 55 | 26 | 0 | | i |
| no | 2 | 1 | 5 | 13 | i |
| no | 65 | 1 | 0 | 8 | i |
| no | 15 | 7 | 12 | 9 | i |
| no | 30 | 15 | 18 | 10 | i |
| no | 63 | 8 | 20 | 11 | i |
| no | 47 | 22 | 25 | 9 | i |
| c | 23 | 12 | 13 | 12 | i |
| c | 38 | 19 | 0 | 14 | i |
| c | 45 | 20 | 21 | 15 | i |
| c | 56 | 25 | 26 | 16 | i |

fault-vt table

| fault no | 1 | 2 | 3 | 6 | 4 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| vt5 | -7.204e-01 | -2.800e+01 | -7.204e-01 | -7.204e-01 | -7.204e-01 | -7.204e-01 | -4.891e+00 | -7.491e-01 | -7.204e-01 | -7.204e-01 |
| vt7 | 1.146e+01 | 2.415e+01 | 1.323e+01 | 5.844e+00 | 1.186e+01 | 1.144e+01 | 2.415e+01 | 1.147e+01 | 3.622e+00 | [illegible] |
| vt8 | 2.625e+01 | 2.515e+01 | 2.763e+01 | 2.730e+01 | 2.667e+01 | 2.632e+01 | 2.515e+01 | 2.625e+01 | 2.719e+01 | [illegible] |
| vt12 | 1.063e+01 | 2.311e+01 | 3.791e-12 | 2.791e+00 | 1.103e+01 | 1.058e+01 | 2.311e+01 | 1.064e+01 | 3.622e+00 | [illegible] |
| vt13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| vt15 | 1.060e+01 | 2.200e+01 | 5.546e-12 | 2.785e+00 | 1.103e+01 | 1.055e+01 | 2.200e+01 | 1.061e+01 | 3.614e+00 | 8.717e+00 |
| vt18 | 9.813e+00 | 1.795e+01 | -5.396e-10 | 2.067e+00 | 1.668e-12 | 9.765e+00 | 1.795e+01 | 9.826e+00 | 2.989e+00 | 8.717e+00 |
| vt19 | 6.927e+00 | 1.267e+01 | -3.809e-10 | 1.459e+00 | 1.154e-12 | 6.893e+00 | 1.267e+01 | 6.936e+00 | 2.035e+00 | 6.153e+00 |
| vt20 | 2.056e+01 | 1.795e+01 | 2.763e+01 | 2.610e+01 | 2.667e+01 | 2.066e+01 | 1.795e+01 | 2.625e+01 | 2.551e+01 | 2.632e+01 |
| vt22 | 1.061e+01 | 2.308e+01 | 3.126e-12 | 2.788e+00 | 1.101e+01 | 9.873e+00 | 2.308e+01 | 1.062e+01 | 3.618e+00 | 1.039e+01 |
| vt25 | 9.867e+00 | 2.228e+01 | 4.915e-10 | 2.079e+00 | 1.026e+01 | 7.040e+00 | 2.228e+01 | 9.880e+00 | 2.905e+00 | 5.652e+00 |
| vt26 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| fault no | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|
| vt5 | -7.204e-01 | -7.204e-01 | -7.204e-01 | -7.204e-01 | -7.204e-01 |
| vt7 | 1.136e+01 | 4.892e+00 | 1.065e+01 | 9.703e+00 | [illegible] |
| vt8 | 2.627e+01 | 2.224e+01 | 2.543e+01 | 2.465e+01 | 2.124e+01 |
| vt12 | 1.849e+01 | 3.208e+00 | 9.825e+00 | 8.820e+00 | 5.795e+00 |
| vt13 | 0 | 3.208e+00 | 0 | 0 | 0 |
| vt15 | 1.046e+01 | 3.201e+00 | 9.697e+00 | 7.147e+00 | 5.780e+00 |
| vt18 | 9.677e+00 | 2.480e+00 | 8.607e+00 | 1.377e+00 | 5.037e+00 |
| vt19 | 6.831e+00 | 1.750e+00 | 3.129e-14 | 9.718e-01 | 3.555e+00 |
| vt20 | 2.067e+01 | 2.080e+01 | 8.607e+00 | 1.377e+00 | 1.632e+01 |
| vt22 | 9.534e+00 | 3.205e+00 | 9.812e+00 | 8.808e+00 | 5.435e+00 |
| vt25 | 9.534e+00 | 2.493e+00 | 9.072e+00 | 8.872e+00 | 3.647e+00 |
| vt26 | 0 | 0 | 0 | 0 | 3.447e+00 |

a new video amplifier circuit

| branch name | no | from node | to node | control branch | value | fault no | |
|---|---|---|---|---|---|---|---|
| e | 7 | 6 | 0 | | -2.800e+01 | | e |
| e | 14 | 28 | 0 | | 2.800e+01 | | e |
| e | 5 | 4 | 5 | | 7.000e-01 | | e |
| e | 19 | 11 | 12 | | 7.000e-01 | | e |
| e | 34 | 17 | 18 | | 7.000e-01 | | e |
| e | 51 | 24 | 25 | | 7.000e-01 | | e |
| nc | 1 | 1 | 2 | | | 2 | e |
| nc | 16 | 7 | 9 | | | 3 | e |
| nc | 26 | 14 | 8 | | | 6 | e |
| nc | 31 | 15 | 29 | | | 4 | e |
| nc | 48 | 22 | 30 | | | 0 | e |
| nc | 60 | 27 | 8 | | | 7 | e |
| d1 | 4 | 4 | 3 | | | | e |
| d2 | 10 | 7 | 5 | | | | e |
| d3 | 18 | 11 | 10 | | | | e |
| d4 | 25 | 14 | 12 | | | | e |
| d5 | 33 | 17 | 16 | | | | e |
| d6 | 42 | 20 | 18 | | | | e |
| d7 | 50 | 24 | 23 | | | | e |
| d8 | 58 | 27 | 25 | | | | e |
| r | 61 | 1 | 0 | | 1.200e+03 | | r |
| r | 3 | 2 | 3 | | 5.000e+02 | | r |
| r | 6 | 5 | 6 | | 5.670e+03 | | r |
| r | 12 | 7 | 8 | | 3.000e+03 | | r |
| r | 17 | 9 | 10 | | 1.000e+03 | | r |
| r | 20 | 12 | 0 | | 1.200e+03 | | r |
| r | 21 | 13 | 0 | | 5.000e+01 | | r |
| r | 13 | 28 | 8 | | 7.800e+01 | | r |
| r | 28 | 12 | 15 | | 3.300e+02 | | r |
| r | 39 | 12 | 22 | | 3.300e+02 | | r |
| r | 32 | 29 | 16 | | 1.000e+03 | | r |
| r | 35 | 18 | 19 | | 5.000e+02 | | r |
| r | 37 | 19 | 0 | | 1.200e+03 | | r |
| r | 40 | 20 | 8 | | 1.000e+03 | | r |
| r | 44 | 21 | 0 | | 5.000e+01 | | r |
| r | 49 | 30 | 23 | | 1.000e+03 | | r |
| r | 53 | 25 | 0 | | 3.300e+03 | | r |
| r | 54 | 26 | 0 | | 5.000e+01 | | r |
| r | 66 | 1 | 2 | | 1.000e+13 | | r |
| r | 67 | 7 | 9 | | 1.000e+13 | | r |
| r | 68 | 15 | 29 | | 1.000e+13 | | r |
| r | 69 | 23 | 24 | | 1.000e+10 | | r |
| cc | 9 | 7 | 5 | 3 | 4.000e+02 | | cc |
| cc | 24 | 14 | 12 | 17 | 6.700e+01 | | cc |
| cc | 41 | 20 | 18 | 32 | 6.700e+01 | | cc |
| cc | 57 | 27 | 25 | 49 | 6.700e+01 | | cc |
| vt5 | 8 | 5 | 0 | | | | i |
| vt7 | 11 | 7 | 0 | | | | i |
| vt8 | 59 | 8 | 0 | | | | i |
| vt12 | 27 | 12 | 0 | | | | i |
| vt13 | 22 | 13 | 0 | | | | i |
| vt15 | 29 | 15 | 0 | | | | i |
| vt18 | 43 | 18 | 0 | | | | i |
| vt19 | 36 | 19 | 0 | | | | i |
| vt20 | 64 | 20 | 0 | | | | i |

| | | | | | |
|---|---|---|---|---|---|
| vt22 | 46 | 22 | 0 | | 1 |
| vt25 | 52 | 25 | 0 | | 1 |
| vt26 | 55 | 26 | 0 | | 1 |
| no | 2 | 1 | 5 | 8 | 1 |
| no | 65 | 1 | 0 | 9 | 1 |
| no | 15 | 7 | 12 | 10 | 1 |
| no | 30 | 15 | 18 | 11 | 1 |
| no | 63 | 8 | 20 | 9 | 1 |
| no | 47 | 22 | 25 | 12 | 1 |
| c | 23 | 12 | 13 | 13 | 1 |
| c | 38 | 19 | 0 | 14 | 1 |
| c | 45 | 20 | 21 | 15 | 1 |
| c | 56 | 25 | 26 | 16 | 1 |

0

| | diode current | diode voltage |
|---|---|---|
| d1 | 1.200e-05 | 0 |
| d2 | 0 | 1.218e+01 |
| d3 | 1.321e-04 | 0 |
| d4 | 0 | 1.563e+01 |
| d5 | 8.488e-05 | 0 |
| d6 | 0 | 1.075e+01 |
| d7 | 4.397e-05 | 0 |
| d8 | 0 | 1.638e+01 |

nominal test node voltages

| node | voltage |
|---|---|
| vt5 | -7.204e-01 |
| vt7 | 1.146e+01 |
| vt8 | 2.625e+01 |
| vt12 | 1.063e+01 |
| vt13 | 0 |
| vt15 | 1.060e+01 |
| vt18 | 9.813e+00 |
| vt19 | 6.927e+00 |
| vt20 | 2.056e+01 |
| vt22 | 1.061e+01 |

| | |
|---|---|
| vt25 | 9.867e+00 |
| vt26 | 0 |

node no: vt5

| set | center | from | to | set code |
|---|---|---|---|---|
| 1 | 1 | -1.120e+00 | -3.204e-01 | 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 |
| 2 | 2 | -2.840e+01 | -2.760e+01 | 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 3 | 8 | -5.291e+00 | -4.491e+00 | 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 |

node no: vt7

| set | center | from | to | set code |
|---|---|---|---|---|
| 1 | 1 | 1.106e+01 | 1.166e+01 | 1 0 0 0 0 1 0 1 0 1 1 0 0 0 0 |
| 2 | 2 | 2.375e+01 | 2.455e+01 | 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 |
| 3 | 3 | 1.283e+01 | 1.363e+01 | 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 |
| 4 | 6 | 5.444e+00 | 6.213e+00 | 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 |
| 5 | 4 | 1.166e+01 | 1.226e+01 | 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 |
| 6 | 10 | 3.222e+00 | 4.022e+00 | 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 |
| 7 | 13 | 4.492e+00 | 5.292e+00 | 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 |
| 8 | 14 | 1.025e+01 | 1.105e+01 | 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 |
| 9 | 15 | 9.303e+00 | 1.010e+01 | 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 |
| 10 | 16 | 6.213e+00 | 6.982e+00 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 |

node no: vt8

| set | center | from | to | set code |
|---|---|---|---|---|
| 1 | 1 | 2.585e+01 | 2.646e+01 | 1 0 0 0 0 1 0 1 0 1 1 0 0 0 0 |
| 2 | 2 | 2.490e+01 | 2.555e+01 | 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 |
| 3 | 3 | 2.741e+01 | 2.803e+01 | 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 |
| 4 | 4 | 2.646e+01 | 2.693e+01 | 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 |
| 5 | 10 | 2.693e+01 | 2.741e+01 | 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 |
| 6 | 13 | 2.184e+01 | 2.264e+01 | 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 |

| | | | | |
|---|---|---|---|---|
| 7 | 15 | 2.425e+01 | 2.490e+01 | 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 |
| 8 | 16 | 2.084e+01 | 2.164e+01 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 |

node no: vt12

| set | center | from | to | set code |
|---|---|---|---|---|
| 1 | 1 | 1.023e+01 | 1.083e+01 | 1 0 0 0 0 1 0 1 0 1 1 0 0 0 0 |
| 2 | 2 | 2.271e+01 | 2.351e+01 | 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 |
| 3 | 3 | -4.000e-01 | 4.000e-01 | 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 |
| 4 | 6 | 2.391e+00 | 3.000e+00 | 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 |
| 5 | 4 | 1.083e+01 | 1.143e+01 | 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 |
| 6 | 10 | 3.415e+00 | 4.022e+00 | 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 |
| 7 | 13 | 3.000e+00 | 3.415e+00 | 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 |
| 8 | 14 | 9.425e+00 | 1.023e+01 | 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 |
| 9 | 15 | 8.420e+00 | 9.220e+00 | 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 |
| 10 | 16 | 5.395e+00 | 6.195e+00 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 |

node no: vt13

| set | center | from | to | set code |
|---|---|---|---|---|
| 1 | 1 | -4.000e-01 | 4.000e-01 | 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 |
| 2 | 13 | 2.808e+00 | 3.608e+00 | 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 |

node no: vt15

| set | center | from | to | set code |
|---|---|---|---|---|
| 1 | 1 | 1.020e+01 | 1.081e+01 | 1 0 0 0 0 1 0 1 0 0 1 0 0 0 0 |
| 2 | 2 | 2.160e+01 | 2.240e+01 | 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 |
| 3 | 3 | -4.000e-01 | 4.000e-01 | 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 |
| 4 | 6 | 2.385e+00 | 2.993e+00 | 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 |
| 5 | 4 | 1.081e+01 | 1.143e+01 | 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 |
| 6 | 10 | 3.408e+00 | 4.014e+00 | 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 |
| 7 | 11 | 8.317e+00 | 9.117e+00 | 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 |

| | | | | |
|---|---|---|---|---|
| 8 | 13 | 2.993e+00 | 3.408e+00 | 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 |
| 9 | 14 | 9.297e+00 | 1.010e+01 | 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 |
| 10 | 15 | 6.747e+00 | 7.547e+00 | 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 |
| 11 | 16 | 5.380e+00 | 6.180e+00 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 |

node no: vt18

| set | center | from | to | set code |
|---|---|---|---|---|
| 1 | 1 | 9.413e+00 | 1.021e+01 | 1 0 0 0 0 1 0 1 0 0 1 0 0 0 0 |
| 2 | 2 | 1.755e+01 | 1.835e+01 | 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 |
| 3 | 3 | -4.000e-01 | 4.000e-01 | 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 |
| 4 | 6 | 1.722e+00 | 2.273e+00 | 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 |
| 5 | 10 | 2.684e+00 | 3.289e+00 | 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 |
| 6 | 11 | 8.317e+00 | 9.117e+00 | 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 |
| 7 | 13 | 2.273e+00 | 2.684e+00 | 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 |
| 8 | 15 | 9.767e-01 | 1.722e+00 | 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 |
| 9 | 16 | 4.637e+00 | 5.437e+00 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 |

node no: vt19

| set | center | from | to | set code |
|---|---|---|---|---|
| 1 | 1 | 6.540e+00 | 7.327e+00 | 1 0 0 0 0 1 0 1 0 0 1 0 0 0 0 |
| 2 | 2 | 1.227e+01 | 1.307e+01 | 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 |
| 3 | 3 | -4.000e-01 | 4.000e-01 | 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 |
| 4 | 6 | 1.215e+00 | 1.749e+00 | 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 |
| 5 | 10 | 1.749e+00 | 2.439e+00 | 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 |
| 6 | 11 | 5.753e+00 | 6.540e+00 | 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 |
| 7 | 15 | 5.718e-01 | 1.215e+00 | 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 |
| 8 | 16 | 3.155e+00 | 3.955e+00 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 |

node no: vt20

| set | center | from | to | set code |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 2.016e+01 | 2.096e+01 | 1 0 0 0 0 1 0 0 0 0 1 1 0 0 0 |
| 2 | 2 | 1.755e+01 | 1.835e+01 | 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 |
| 3 | 3 | 2.723e+01 | 2.803e+01 | 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 |
| 4 | 6 | 2.581e+01 | 2.638e+01 | 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 |
| 5 | 4 | 2.638e+01 | 2.707e+01 | 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 |
| 6 | 10 | 2.511e+01 | 2.581e+01 | 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 |
| 7 | 14 | 8.207e+00 | 9.007e+00 | 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 |
| 8 | 15 | 9.767e-01 | 1.777e+00 | 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 |

node no: vt22

| set | center | from | to | set code |
|---|---|---|---|---|
| 1 | 1 | 1.024e+01 | 1.101e+01 | 1 0 0 0 1 0 0 1 0 1 0 0 0 0 0 |
| 2 | 2 | 2.268e+01 | 2.348e+01 | 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 |
| 3 | 3 | -4.000e-01 | 4.000e-01 | 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 |
| 4 | 6 | 2.388e+00 | 2.996e+00 | 0 0 0 1 0 0 0 0 0 0 0 0 0 0 |
| 5 | 7 | 9.473e+00 | 1.024e+01 | 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 |
| 6 | 10 | 3.411e+00 | 4.018e+00 | 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 |
| 7 | 13 | 2.996e+00 | 3.411e+00 | 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 |
| 8 | 15 | 8.408e+00 | 9.208e+00 | 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 |
| 9 | 16 | 5.035e+00 | 5.835e+00 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 |

node no: vt25

| set | center | from | to | set code |
|---|---|---|---|---|
| 1 | 1 | 9.469e+00 | 1.027e+01 | 1 0 0 0 1 0 0 1 0 1 1 0 0 0 0 |
| 2 | 2 | 2.188e+01 | 2.268e+01 | 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 |
| 3 | 3 | -4.000e-01 | 4.000e-01 | 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 |
| 4 | 6 | 1.679e+00 | 2.286e+00 | 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 |
| 5 | 7 | 6.640e+00 | 7.440e+00 | 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 |
| 6 | 10 | 2.699e+00 | 3.276e+00 | 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 |
| 7 | 13 | 2.286e+00 | 2.699e+00 | 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 |

| | | | | |
|---|---|---|---|---|
| 8 | 14 | 8.672e+00 | 9.459e+00 | 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 |
| 9 | 15 | 7.672e+00 | 8.472e+00 | 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 |
| 10 | 16 | 3.276e+00 | 4.047e+00 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 |

node no: vt26

| set | center | from | to | set code |
|---|---|---|---|---|
| 1 | 1 | -4.000e-01 | 4.000e-01 | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 |
| 2 | 16 | 3.247e+00 | 4.047e+00 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 |

| fault | fault code | | | | |
|---|---|---|---|---|---|
| | vt5 | vt7 | vt8 | vt12 | vt13 |
| f 1 | 1 | 1 | 1 | 1 | 1 |
| f 9 | 1 | 1 | 1 | 4 | 1 |
| f12 | 1 | 1 | 5 | 1 | 1 |
| f 7 | 1 | 5 | 5 | 1 | 1 |
| f 2 | 2 | 2 | 2 | 2 | 2 |
| f 8 | 2 | 2 | 2 | 2 | 3 |
| f 3 | 3 | 3 | 3 | 3 | 1 |
| f 6 | 4 | 4 | 4 | 4 | 1 |
| f 4 | 5 | 1 | 1 | 5 | 1 |
| f10 | 6 | 6 | 6 | 6 | 1 |
| f11 | 7 | 1 | 1 | 4 | 1 |
| f13 | 8 | 7 | 7 | 1 | 1 |
| f14 | 9 | 8 | 5 | 7 | 1 |
| f15 | 10 | 9 | 8 | 8 | 1 |
| f16 | 11 | 10 | 9 | 2 | 1 |

#eor
#eof

**Appendix 1**

```
c-----------------------------------------------------------------------
c
c       program cctl for accepting network topology for fault simul-
c       tion and producing a special format output file containing
c       the network information.  The output file name is specified
c       by the user.  To be compatible with the "hafdic" program,
c       the output file should be named "data".
c
c       internal variables
c
c       ne      number of independent voltage sources
c       nl      number of inductances
c       nnc     number of normally closed switches
c       nd      number if ideal diodes
c       nr      number of resistances
c       ng      number of conductances
c       ncc     number of current controlled current sources
c       ncv     number of current controlled voltage sources
c       nvc     number of voltage controlled current sources
c       nvv     number of voltage controlled voltage sources
c               nvt     number of test nodes
c       nno     number of normaly open switches
c       nc      number of capacitances
c       nj      number of independent current sources
c       br      vector containing branch numbers
c       nft     vector containing fault numbers
c       bat     vector containing batterry values
c       cur     vector containing values of independent current sources
c       nfrom   vector containing source nodes of the corresponding
c               branches
c       nto     vector containing destination nodes of the correspond-
c               ing branches
c       icont   array containing control branches.
c       x       array containing values of resistances,conductances
c               and control branches.
c       title   array containing problem title
c       ch      array containing branh names
c       filnam  character variable containing file name specified by
c               the user for storing data
c
c       limitations
c
c       40 nodes
c       90 branches
c       40 voltage independent sources
c       40 current independent sources
c       40 dependent sources (all kinds)
c       60 resistances or cinductances
c
c
c-----------------------------------------------------------------------
c
        integer nft(90),br(90)
        dimension bat(30),cur(30),nfrom(90),nto(90),icont(4,90)
        dimension x(6,40)
        character*80 title
        character*4 istop,ch(90)
        character*20 filnam
        character*1 ie,ij,ii,ir,ig,il,id,ic
        character*2 inc,ivt,ino,icc,icv,ivc,ivv
```

```
      data icc,icv,ivc,ivv/'cc','cv','vc','vv'/
      data istop,ie,ij,ii,ir,ig/'stop','e','j','i','r','g'/
      data il,inc,id,ivt,ino,ic/'L','nc','d','vt','no','c'/
c
      ne=0
      nl=0
      nnc=0
      nd=0
      nr=0
      ng=0
      ncc=0
      ncv=0
      nvc=0
      nvv=0
      nvt=0
      nno=0
      nc=0
      nj=0
      np=0
      print*,'problem title:'
      read(5,'(a80)') title
c
      do 5 i=1,90
    5 nft(i)=0
c
      do 100 k=1,90
c
      print*,'branch type(e,j,r,g,cc,cv,vc,vv,vt,L,c,nc,no,d):'
      read(5,'(a4)') ch(k)
      if(ch(k).eq.istop) go to 110
      if(ch(k)(1:1).eq.ie) then
       ne=ne+1
       np=np+1
       print*,'branch no,from,to,battery value'
       read*, br(k),nfrom(k),nto(k),bat(ne)
      elseif(ch(k)(1:1).eq.il) then
       nl=nl+1
       np=np+1
      print*,'branch no,from,to,fault no(if o.c is to be simulated)'
      read*,br(k),nfrom(k),nto(k),nft(np)
      elseif(ch(k)(1:2).eq.inc) then
      nnc=nnc+1
      np=np+1
      print*,'branch no,from,to,fault no(if o.c is to be simulated)'
      read*,br(k),nfrom(k),nto(k),nft(np)
      elseif(ch(k)(1:1).eq.id) then
      nd=nd+1
      np=np+1
      print*,'branch no,from,to'
      read*,br(k),nfrom(k),nto(k)
      elseif(ch(k)(1:1).eq.ir) then
      print*,'branch no,from,to,resistance value'
      nr=nr+1
      read*,br(k),nfrom(k),nto(k),x(1,nr)
      elseif(ch(k)(1:1).eq.ig) then
      print*,'branch no,from,to,conductance value'
      ng=ng+1
      read*,br(k),nfrom(k),nto(k),x(2,ng)
      elseif(ch(k)(1:2).eq.icc) then
      print*,'branch no,from,to,control branch,cc value'
```

```
      ncc=ncc+1
      read*,br(k),nfrom(k),nto(k),icont(1,ncc),x(3,ncc)
      elseif(ch(k)(1:2).eq.icv) then
      print*,'branch no,from,to,control branch,cv value'
      ncv=ncv+1
      read*,br(k),nfrom(k),nto(k),icont(2,ncv),x(4,ncv)
      elseif(ch(k)(1:2).eq.ivc) then
      print*,'branch no,from,to,control branch,vc value'
      nvc=nvc+1
      read*,br(k),nfrom(k),nto(k),icont(3,nvc),x(5,nvc)
      elseif(ch(k)(1:2).eq.ivv) then
      print*,'branch no,from,to,control branch,vv value'
      nvv=nvv+1
      read*,br(k),nfrom(k),nto(k),icont(4,nvv),x(6,nvv)
      elseif(ch(k)(1:2).eq.ivt) then
      nvt=nvt+1
      np=np+1
      print*,'branch no,from,to'
      read*,br(k),nfrom(k),nto(k)
      elseif(ch(k)(1:2).eq.ino) then
      nno=nno+1
      np=np+1
      print*,'branch no,from,to,fault no(if s.c is to be simulated)'
      read*,br(k),nfrom(k),nto(k),nft(np)
      elseif(ch(k)(1:1).eq.ic.and.ch(k)(1:2).ne.icc.and.ch(k)(1:2).ne.
     + icv)then
      nc=nc+1
      np=np+1
      print*,'branch no,from,to,fault no(if s.c is to be simulated)'
      read*,br(k),nfrom(k),nto(k),nft(np)
      elseif(ch(k)(1:1).eq.ij) then
      nj=nj+1
      np=np+1
      print*,'branch no,from,to,value of current source'
      read*,br(k),nfrom(k),nto(k),cur(nj)
      else
       go to 100
      endif
100   continue
110   print*,"ambig set range :"
      read*,amr
      print*,'file name where data is to be stored:'
      read(5,'(a20)') filnam
      open(unit=7,file=filnam,status='new')
      rewind 7
      write(7,'("the first two lines are for program use, DO NOT dele
     + te them")')
      write(7,'(14i3,e10.3)') ne,nl,nnc,nd,nr,ng,ncc,ncv,nvc,nvv,nvt,
     + nno,nc,nj,amr
      write(7,'(a80)') title
      write(7,'(" branch  from  to  control  value  fault  for
     + program")')
      write(7,'("type  no  node node  branch         no.    use
     + only")')
      np=0
      ne=0
      do 120 i=1,k
      if(ch(i)(1:1).eq.ie) then
      ne=ne+1
      np=np+1
```

```
      write(7,250) ch(i),br(i),nfrom(i),nto(i),pat(ne),ie
      endif
120   continue
      nl=0
      do 130 i=1,k
      if(ch(i)(1:1).eq.il) then
      nl=nl+1
      np=np+1
      write(7,260) ch(i),br(i),nfrom(i),nto(i),nft(np),ie
      endif
130   continue
      nnc=0
      do 140 i=1,k
      if(ch(i)(1:2).eq.inc) then
      nnc=nnc+1
      np=np+1
      write(7,260) ch(i),br(i),nfrom(i),nto(i),nft(np),ie
      endif
140   continue
      nd=0
      do 150 i=1,k
      if(ch(i)(1:1).eq.id) then
      nd=nd+1
      np=np+1
      write(7,270) ch(i),br(i),nfrom(i),nto(i),ie
      endif
150   continue
      nr=0
      do 160 i=1,k
      if(ch(i)(1:1).eq.ir) then
      nr=nr+1
      write(7,250) ch(i),br(i),nfrom(i),nto(i),x(1,nr),ir
      endif
160   continue
      ng=0
      do 170 i=1,k
      if(ch(i)(1:1).ne.ig) go to 170
      ng=ng+1
      write(7,250) ch(i),br(i),nfrom(i),nto(i),x(2,ng),ig
170   continue
      ncc=0
      do 180 i=1,k
      if(ch(i)(1:2).ne.icc) go to 180
      ncc=ncc+1
      write(7,280) ch(i),br(i),nfrom(i),nto(i),icont(1,ncc),x(3,ncc),icc
180   continue
      ncv=0
      do 190 i=1,k
      if(ch(i)(1:2).ne.icv) go to 190
      ncv=ncv+1
      write(7,280) ch(i),br(i),nfrom(i),nto(i),icont(2,ncv),x(4,ncv),icv
190   continue
      nvc=0
      do 200 i=1,k
      if(ch(i)(1:2).ne.ivc) go to 200
      nvc=nvc+1
      write(7,280) ch(i),br(i),nfrom(i),nto(i),icont(3,nvc),x(5,nvc),ivc
200   continue
      nvv=0
      do 205 i=1,k
```

```
      if(ch(i)(1:2).ne.ivv) go to 205
      nvv=nvv+1
      write(7,280) ch(i),br(i),nfrom(i),nto(i),icont(6,nvv),x(6,nvv),ivv
205   continue
      nvt=0
      do 210 i=1,k
      if(ch(i)(1:2).ne.ivt) go to 210
      nvt=nvt+1
      np=np+1
      write(7,270) ch(i),br(i),nfrom(i),nto(i),ii
210   continue
      nno=0
      do 215 i=1,k
      if(ch(i)(1:2).ne.ino) go to 215
      nno=nno+1
      np=np+1
      write(7,260) ch(i),br(i),nfrom(i),nto(i),nft(np),ii
215   continue
      nc=0
      do 220 i=1,k
      if(ch(i)(1:1).ne.ic) go to 220
      if(ch(i)(1:2).eq.icc.or.ch(i)(1:2).eq.icv) go to 220
      nc=nc+1
      np=np+1
      write(7,260) ch(i),br(i),nfrom(i),nto(i),nft(np),ii
220   continue
      nj=0
      do 225 i=1,k
      if(ch(i)(1:1).ne.ij) go to 225
      nj=nj+1
      np=np+1
      write(7,250) ch(i),br(i),nfrom(i),nto(i),cur(nj),ii
225   continue
250   format(a4,i4,2i5,7x,e10.3,9x,a2)
260   format(a4,i4,2i5,17x,i3,6x,a2)
270   format(a4,i4,2i5,26x,a2)
280   format(a4,i4,3i5,2x,e10.3,9x,a2)
      stop
      end
```

## Appendix 2

```
c-------------------------------------------------------------------
c
c     progam : hafdic
c
c
c     hafdic is a program for simulating hard faults in piecewise
c     linear analog circuits and generating a fault dictionary.
c     fault simualtion is accomplished by a complemntary pivot alg-
c     orithm for solving a subset of the circuit equilibrium equat-
c     ions, which are formulated only once in the beginning of the
c     program as detailed in vol.1 of "fault diagnosis of nonlinear
c     analog circuits".  a dictionary is finally contained in two
c     tables of testnode voltage ranges and numeric fault codes.
c
c     system: cdc-6600, dual mace, mnf compiler.
c
c        programmer: yassin elcherif
c
c-----------------------------------------------------------------
c
c     input data:
c
c     a special format file "data" must be available in the pfiles
c     storage of the same user id specified in the batch job card.
c     see program "cctl" for preparing the data file.
c
c     internal variables
c
c     bat      vector containing battery values
c     vt       vector containing test node voltages for the currently
c               simulated fault.
c     cur      vector containing values of current sources.
c     nft      vector containing fault numbers
c     header   vector containing the problem title
c     br       vector containing branch numbers
c     type     vector containing branch types
c     nfrom    vector containing the source node of the corresponding
c              branch
c     nto      vector containing the destiny node of the corresponding
c              branch
c     value    vector containing element values
c     icont    vector containing control branches of dependent sources
c     ch       array containing branch names (4 characters each)
c     ksim     vector containing port branches to be simulated as faulty
c              under the same fault number (multiple fault cases).
c     isim     vector flag for to identify faults that have been simulated
c
c     limitations
c     max number of nodes              40
c     max number of brances            90
c     max number of batteries          30
c     max number of independent cuurent sources 30
c     max number of faults             30
c     max number of diodes             50
c     max number of test nodes         30
c     max number of faults to be simultaneously simulated 10
c
c     error messages
c
c     there are two cases when a fault cannot be simulated .
```

```
c     in the first case a zero pivot is encountered in reducing the
c     equations to formulate the complementary problem.  the fault
c     will be skipped and the following error messege will be printed:
c     "fault no:[x]
c      zero pivot-nonzero col,complementary problem cannot be formulated"
c     in the second case the lemke complementary pivot algorithm will
c     terminate in a ray.  the fault will be skipped and the following
c     messege will be printed:
c     "fault no:[x]
c       no complementary solution iteration no[x]"
c
c     debugging
c
c     to print out the complementary problem and the diode currents
c     and voltages for every fault simulation set the variable
c     ipdebug=1 in line 260
c     to print the same information for the nominal case only
c     set ipfl=1 in line 195
c
c-----------------------------------------------------------------
c
      program hafdic(data,tape5=data,output,tape6=output)
c
      dimension bat(30),vt(30),nft(90),cur(30)
      common/004/am,q,w,z,l1,b,nl1,nl2,a,ne1,ne2,ir,mbasis,izr
      dimension am(50,50),b(50,50),q(50),a(50),mbasis(100)
      dimension w(50),z(50)
      dimension nff(30)
      integer aa,header,br,type
      common aa(40,90),ans(90,180),header(320),nfrom(90),nto(90)
      common br(90),type(90),value(90),icont(90)
      integer cn(2,90),ii,jj,isim(90),ksim(10)
c
      do 1 i=1,90
  1   nft(i)=0
      rewind 5
      read(5,101) (header(i),i=1,80)
      read(5,102) nb,nl,nnc,nd,nr,ng,ncc,ncv,nvc,nvv,nvt,nno,nc,nj,amr
      read(5,101) (header(i),i=1,80)
      write(6,1011) (header(i),i=1,80)
      write(6,107)
      write(6,108)
      read(5,101) (header(i),i=1,80)
      read(5,101) (header(i),i=1,80)
      np=0
      i=1
      while(i.le.nb) do
      np=np+1
      read(5,103) ch(1,i),ch(2,i),br(i),nfrom(i),nto(i),bat(i),type(i)
      write(6,1031) ch(1,i),ch(2,i),br(i),nfrom(i),nto(i),bat(i),type(i)
      i=i+1
      endwhile
      n=np+nl
      while(i.le.n) do
      np=np+1
      read(5,104) ch(1,i),ch(2,i),br(i),nfrom(i),nto(i),nft(np),type(i)
      write(6,1041) ch(1,i),ch(2,i),br(i),nfrom(i),nto(i),nft(np),
     +type(i)
      i=i+1
      endwhile
```

```
      n=n+nnc
      while(i.le.n) do
      np=np+1
      read(5,104) ch(1,i),ch(2,i),br(i),nfrom(i),nto(i),nft(np),type(i)
      write(6,1041) ch(1,i),ch(2,i),br(i),nfrom(i),nto(i),nft(np),
     +type(i)
      i=i+1
      endwhile
      n=n+nd
      while(i.le.n) do
      np=np+1
      read(5,105) ch(1,i),ch(2,i),br(i),nfrom(i),nto(i),type(i)
      write(6,1051) ch(1,i),ch(2,i),br(i),nfrom(i),nto(i),type(i)
      i=i+1
      endwhile
      n=n+nr+ng
      while(i.le.n) do
      read(5,103) ch(1,i),ch(2,i),br(i),nfrom(i),nto(i),value(i),type(i)
      write(6,1031) ch(1,i),ch(2,i),br(i),nfrom(i),nto(i),value(i),
     +type(i)
      i=i+1
      endwhile
      n=n+ncc+ncv+nvc+nvv
      while(i.le.n) do
      read(5,106) ch(1,i),ch(2,i),br(i),nfrom(i),nto(i),icont(i),
     +value(i),type(i)
      write(6,1061) ch(1,i),ch(2,i),br(i),nfrom(i),nto(i),icont(i),
     +value(i),type(i)
      i=i+1
      endwhile
      n=n+nvt
      while(i.le.n) do
      np=np+1
      read(5,105) ch(1,i),ch(2,i),br(i),nfrom(i),nto(i),type(i)
      write(6,1051) ch(1,i),ch(2,i),br(i),nfrom(i),nto(i),type(i)
      i=i+1
      endwhile
      n=n+nno+nc
      while(i.le.n) do
      np=np+1
      read(5,104) ch(1,i),ch(2,i),br(i),nfrom(i),nto(i),nft(np),type(i)
      write(6,1041) ch(1,i),ch(2,i),br(i),nfrom(i),nto(i),nft(np),
     +type(i)
      i=i+1
      endwhile
      n=n+nj
      k=0
      while(i.le.n) do
      np=np+1
      k=k+1
      read(5,103) ch(1,i),ch(2,i),br(i),nfrom(i),nto(i),cur(k),type(i)
      write(6,1031) ch(1,i),ch(2,i),br(i),nfrom(i),ntc(i),cur(k),type(i)
      i=i+1
      endwhile
 101  format(80a1)
 1011 format(1h1,30x,80a1)
 102  format(14i3,e10.3)
 103  format(a2,a2,i4,2i5,7x,e10.3,9x,a2)
 1031 format(2x,a2,a2,i4,2i5,7x,e10.3,9x,a2)
 104  format(a2,a2,i4,2i5,17x,i3,6x,a2)
```

```
 1041 format(2x,a2,a2,i4,2i5,17x,i3,6x,a2)
 105  format(a2,a2,i4,2i5,26x,a2)
 1051 format(2x,a2,a2,i4,2i5,26x,a2)
 106  format(a2,a2,i4,3i5,2x,e10.3,9x,a2)
 1061 format(2x,a2,a2,i4,3i5,2x,e10.3,9x,a2)
 107  format('  branch  from  to  control  value  fault')
 108  format('name  no  node node  branch          no')
c
      ndv=nd+nvt
      n1=nb+nl+nnc
      ntb=np+nr+ng+ncc+ncv+nvc+nvv
c obtain hybrid matrix
c
      call hybrid(ii,jj,0,ntb)
      ipfl=1
c
c
c obtain number of non-port branches
c
      ii=ii-1
      jj=jj-1
c
c form and solve the complementary problem of
c the nominal case
c
      do 4 i=1,ndv
      q(i)=0.0
      do 2 j=1,nd
    2 am(i,j)=ans(i+n1+ii,j+np+n1+jj)
      do 3 j=1,nb
    3 q(i)=q(i)+bat(j)*ans(i+n1+ii,j+np+jj)
      n=n1+nd+nr+ng+ncc+ncv+nvc+nvv+nvt+nno+nc
      if(nj.ne.0) then
      do 303 j=1,nj
  303 q(i)=q(i)+cur(j)*ans(ii+n1+i,jj+np+n+j)
      endif
    4 continue
      if(ipfl.eq.1) go to 201
      write(6,19)
      ndg=0
      l=1
 2012 ndg=min0(nd,ndg+9)
      do 2020 i=1,nd
      if(ndg.lt.nd) then
      write(6,20) (am(i,j),j=l,ndg)
      else
      write(6,20) (am(i,j),j=l,ndg),q(i)
      endif
 2020 continue
      l=l+9
      if(ndg.lt.nd) go to 2012
  201 call lemke(nd)
c zero out vt(i)
c calculate and print out node voltages of the nominal case
c
      do 5 i=1,nvt
    5 vt(i)=0.0
      do 7 i=1,nvt
      vt(i)=vt(i)-q(i+nd)
      do 7 j=1,nd
```

```
    7  vt(i)=vt(i)-am(i+nd,j)*z(j)
       do 16 i=1,nvt
   16  ans(i,1)=vt(i)
       n3=n1
       n4=n1+nd+nr+ng+ncc+ncv+nvc+nvv
       write(6,10)
       write(6,8) (ch(1,i+n3),ch(2,i+n3),w(i),z(i),i=1,nd)
       write(6,11)
       write(6,9) (ch(1,i+n4),ch(2,i+n4),vt(i),i=1,nvt)
   10  format(//10x,'diode current',10x,'diode voltage')
    8  format(/5x,2a2,e10.3,15x,e10.3)
   11  format(//2x'nominal test node voltages'//5x,'node',5x,'voltage')
    9  format(/5x,2a2,4x,e10.3)
   19  format(/2x'complementary problem'//5x,'coeff matrix, last col=q')
   20  format(/10(1x,e10.3))
   18  format(//2x,'no of iterations = ',i3)
       nb1=nb+1
       nnf=0
       zero=1.e-8
       idebug=0
       do 400 i=1,90
  400  isim(i)=0
       nnf=1
       nff(1)=1
c
c loop to simulate faults
c
       do 90 nf=nb1,np
       if(nf.ge.n) go to 90
       if(nft(nf).eq.0) go to 90
       if(isim(nf).eq.1) go to 90
       do 320 i=1,10
  320  ksim(i)=0
       m=0
       do 330 k=nf,np
       if(nft(k).ne.nft(nf)) go to 330
       m=m+1
       ksim(m)=k
       isim(k)=1
  330  continue
       iflag=0
       if(nf.gt.n1.and.nf.le.(n1+ndv)) go to 90
c
c simulate only faults identified in the input
c
  150  do 23 i=1,ndv
       q(i)=0.0
       do 21 j=1,nd
   21  am(i,j)=ans(i+n1+ii,j+np+n1+jj)
       do 22 j=1,nb
   22  q(i)=q(i)+bat(j)*ans(i+n1+ii,j+np+jj)
       n=n1+nd+nr+ng+ncc+ncv+nvc+nvv+nvt+nno+nc
       if(nj.ne.0) then
       do 322 j=1,nj
  322  q(i)=q(i)+cur(j)*ans(ii+n1+i,jj+np+n+j)
       endif
       am(i,nd+1)=q(i)
       do 340 l=1,m
       k=ksim(l)
  340  am(i,nd+1+l)=ans(ii+n1+i,jj+np+k)
```

```
   23  continue
c
c
       ndvm=ndv+m
       ndv1=ndv+1
       do 345 i=ndv1,ndvm
  345  am(i,nd+1)=0.0
c
       do 360 i=ndv1,ndvm
       kf=ksim(i-ndv)
       do 346 j=1,nb
  346  am(i,nd+1)=am(i,nd+1)+bat(j)*ans(ii+kf,jj+np+j)
       do 347 j=1,nj
  347  am(i,nd+1)=am(i,nd+1)+cur(j)*ans(ii+kf,jj+np+n+j)
c
c
       do 348 j=1,nd
  348  am(i,j)=ans(ii+kf,jj+np+n1+j)
c
c
       do 349 l=1,m
       k=ksim(l)
  349  am(i,nd+1+l)=ans(ii+kf,jj+np+k)
  360  continue
       do 390 l=1,m
       pivot=am(ndv+l,nd+1+l)
       if(abs(pivot).lt.zero) then
       do 370 i=1,ndvm
       if(abs(am(i,nd+l+1)).gt.zero) then
       print 361,nft(nf)
  361  format(//'fault no:',i2,'zero pivot, nonzero column in the',
      +'hybrid matrix'/'complementary problem cannot be formulated')
       write(6,362) ((am(ik,jk),jk=1,6),ik=1,6)
  362  format(6(2x,e10.3))
       go to 90
       endif
  370  continue
       if(m.eq.1) then
       nnf=nnf+1
       do 36 i=1,nvt
   36  ans(i,nnf)=ans(i,1)
       go to 90
       endif
       go to 390
       endif
       ndm=nd+m
       do 372 j=1,ndm
  372  am(ndv+l,j)=am(ndv+l,j)/pivot
       do 374 i=1,ndvm
       do 374 j=1,ndm
       if(i.eq.(ndv+l)) go to 374
       am(i,j)=am(i,j)-am(ndv+l,j)*am(i,nd+l+1)
  374  continue
  390  continue
c
       do 28 i=1,ndv
   28  q(i)=am(i,nd+1)
       if(iflag.eq.0) go to 209
       if(idebug.eq.0) go to 209
       print 210
```

```
 210  format(2x,'before attempting to solve by lemke algorithm
     +the problem was')
      go to 211
 209  call lemke(nd)
 211  if(ir.ne.1001) go to 206
      if(idebug.eq.0) go to 206
      if(iflag.eq.1) go to 3011
      write(6,15) nft(nf),nf
      write(6,19)
 3011 l=1
      ndg=0
 3012 ndg=min0(nd,ndg+9)
      do 3020 i=1,nd
      if(i.eq.1) then
      write(6,3013) (ch(1,n3+j),ch(2,n3+j),j=1,nd)
      endif
 3013 format(10(1x,2a2,4x))
      if(ndg.lt.nd) then
      write(6,20) (am(i,j),j=l,ndg)
      else
      write(6,20) (am(i,j),j=l,ndg),q(i)
      endif
 3020 continue
      l=l+9
      if(ndg.lt.nd) go to 3012
      write(6,18) l1
      write(6,10)
  15  format(//2x,'fault case no: ',i2,5x,'column no: ',i2)
      if(iflag.eq.1) go to 206
      if(idebug.eq.0) go to 206
      write(6,8) (ch(1,i+n3),ch(2,i+n3),w(i),z(i),i=1,nd)
      iflag=1
      go to 150
c
c calculate node voltages under fault condition nft(nf)
c
 206  do 35 i=1,nvt
  35  vt(i)=0.0
      do 37 i=1,nvt
      vt(i)=-q(i+nd)
      do 37 j=1,nd
  37  vt(i)=vt(i)-am(i+nd,j)*z(j)
      if(ipfl.eq.1)go to 207
      write(6,38)
  38  format(//5x,'node',5x,'voltage')
      write(6,9) (ch(1,i+n4),ch(2,i+n4),vt(i),i=1,nvt)
c
c store the node voltages of the current fault case in the
c f-vt table
c
 207  if(ir.eq.1001) go to 90
      nnf=nnf+1
      nff(nnf)=nft(nf)
      do 39 i=1,nvt
  39  ans(i,nnf)=vt(i)
  90  continue
c
c print out f-vt table
c
      nfmax=nnf
```

```
      write(6,41)
   41 format(//2x,'fault-vt table')
      l=1
      l10=0
   40 l10=amin0(l10+10,nfmax)
      write(6,42) (nff(j),j=l,l10)
   42 format(//2x,'fault no',2x,10(i2,9x))
      do 43 i=1,nvt
   43 write(6,44) ch(1,i+n4),ch(2,i+n4),(ans(i,j),j=l,l10)
   44 format(//2x,2a2,4x,10(e10.3,1x))
      if(l10.ge.nfmax) go to 111
      l=l+10
      go to 40
  111 call ambset(nvt,nfmax,ch,n4,amr,nff)
      end
#eor
```

```
c-----------------------------------------------------------------------
c
 subroutine hybrid for obtaining the hybrid equations of the
c      n port network.
c
c         Input and common variables
c
c      a         array containing incedence matrix
c      ans       array containing the hybrid matrix in the reduced
c                echelon form
c      nport1    starting row of the hybrid matrix in ans
c      ii        starting column of the hybrid matrix in ans
c      debug     flag for printing debugging outout
c      ntb       total number of branches in the network
c
c-----------------------------------------------------------------------
c
      subroutine hybric(nport1,ii,iprnt,ntb,debug)
      integer a,nfrom(90),nto(90),icont(90),type(90),dcol(90)
      integer icount(2),header(320),br(90),rbr(90),ansrow,anscol
      integer cv,e,r,cc,vv,vc,from,to,c,coun,begin,temp,st,tn
      integer tp,port,debug,istp,g,v
      real value(90),f3(90,40),f6(90,40),ans(90,180)
      common a(40,90),ans,header,nfrom,nto
      common br,type,value,icont
      data e ,is,r,g,vv,cv,cc,vc,st,c,v/2h e,2h i,2h r,2h g,
     12hvv,2hcv,2hcc,2hvc,2hst,1hi,1hv/
      data istp/1hs/
c
c  max circuit configuration is 40 nodes and 80 branches
c  max of 40 elements in any category (tp,tn,ln,lp)
c
1     nbr=0
      nnode=0
c
c
c zero out a matrix
      do 2 i=1,40
      do 2 j=1,90
2     a(i,j)=0
c read in data and fill a matrix
      debug=0
      do 3 k=1,ntb
c  store entries into a matrix
      from=nfrom(k)
      to=nto(k)
      if(from.ne.0)a(from,k)=1
      if(to.ne.0)a(to,k)=-1
      nbr=max0(nbr,k)
3     nnode=max0(nnode,nfrom(k),nto(k))
      zero=1.e-8
      if(debug.ne.1) go to 5
c  print the a matrix for debug run
      write(6,505)
      do 6 i=1,nnode
6     write(6,506)(a(i,j),j=1,nbr)
5     do 7 i=1,nbr
      dcol(i)=0
7     rbr(i)=0
c
```

```
c  determine elements making up the tree
c
       call ftree(nnode,nbr,dcol)
c
c reorder a matrix into four classes
c
c   1. tree port branches(tp)
c   2. tree non-port branches(tn)
c   3. link non-port branches(ln)
c   4. link port branches(lp)
c
c dcol contains ordering of a with tree branches in leftmost columns
       jj=nnode+1
       n=1
       do 8 j=1,nnode
       m=dcol(j)
       do 9 k=n,m
       if(m.eq.k)go to 8
       dcol(jj)=k
       jj=jj+1
9      continue
8      n=m+1
       if(jj.eq.ncol)go to 10
       do 11 i=n,nbr
11     dcol(i)=i
c reorder dcol into four classes
c icount(1) marks last port column of tree branches
c icount(2) marks last non-port column of link branches
10     icount(1)=1
       it2=nnode
       i=1
13     do 12 m=1,it2
       mm=(nbr+1)*(i-1)+((3-(2*i))*m)
       item=dcol(mm)
       if(type(item).ne.e,and.type(item).ne.is)go to 12
       item1=icount(i)
       dcol(mm)=dcol(item1)
       dcol(item1)=item
       icount(i)=icount(i)+1-((i-1)*2)
12     continue
       if(i.eq.2) go to 14
       icount(1)=icount(1)-1
       icount(2)=nbr
       it2=nbr-nnode
       i=2
       go to 13
c reorder the a matrix and the original label vector to correspond to
c  the reordered dcol
14     nn=2
       n=1
       begin=1
       coun=0
15     item=dcol(n)
       if(item.eq.begin)go to 16
       itemp=br(n)
       br(n)=br(item)
       br(item)=itemp
       do 17 j=1,nnode
       temp=a(j,n)
       a(j,n)=a(j,item)
```

```
17    a(j,item)=temp
      coun=coun+1
      dcol(n)=-dcol(n)
      n=item
      go to 15
16    dcol(n)=-dcol(n)
      if(coun.eq.(nbr-1))go to 18
      do 19 i=nn,nbr
      item=dcol(i)
      if(item.eq.i)go to 20
      if(item.lt.0)go to 19
      begin=i
      n=i
      go to 15
20    coun=coun+1
      dcol(i)=-dcol(i)
      nn=i
19    continue
18    do 22 n=1,nbr
22    dcol(n)=iabs(dcol(n))
c
c  reduce reordered a matrix to row echelon form
c
      call iaech(nnode,nbr)
c
c  back substitute a matrix
c
      do 23 i=2,nnode
      lrow=i-1
      do 23 j=1,lrow
      ifcol=i
      itemp=a(j,ifcol)
      do 23 k=i,nbr
23    a(j,k)=a(j,k)-a(i,k)*itemp
c
c formulate the element characteristics
c
c tp is number of columns in f1 and f5
c tn is number of columns in f2 and f6
c ln is number of columns in f3 and f7
c lp is number of columns in f4 and f8
      tp=icount(1)
      tn=nnode-icount(1)
      ln=icount(2)-nnode
      lp=nbr-icount(2)
      port=tp+lp
      nport=tn+ln
      ansrow=nbr
      anscol=nbr+port
      write(6,507)
      if(tp.eq.0) go to 24
      write(6,508) (br(i),i=1,tp)
24    j=tp+1
      if(tn.eq.0) go to 25
      write(6,509) (br(i),i=j,nnode)
25    j=nnode+1
      jj=nnode+ln
      if(ln.eq.0) go to 26
      write(6,510) (br(i),i=j,jj)
26    j=jj+1
```

```
      if(lp.eq.0) go to 28
      write(6,511) (br(i),i=j,nbr)
c  zero ans matrix
28    do 27 i=1,ansrow
      do 27 j=1,anscol
27    ans(i,j)=0.0
      do 29 i=1,nport
      do 30 j=1,tn
30    f6(i,j)=0
      do 29 j=1,ln
29    f3(i,j)=0
      kount=icount(1)
      k=0
      j=1
      do 31 i=1,nbr
      item=br(i)
31    rbr(item)=i
      if(debug.ne.1) go to 32
      write(6,512) tp,tn,ln,lp
      write(6,513)(br(i),i=1,nbr)
32    kount=kount+1
      mm=dcol(kount)
      itemp=icont(mm)
      itemp=rbr(itemp)
      it1=port+j
      if(type(mm).eq.g.or.type(mm).eq.vc.or.type(mm).eq.cc)
     1go to 33
c voltage source type
      if (kount.gt.nnode)go to 34
c  f2
      it2=ln+j
      ans(it1,it2)=1.
      if(type(mm).eq.cv)go to 35
      if(type(mm).eq.vv)go to 36
      f6(j,j)=-value(mm)
      go to 37
34    k=k+1
      f3(j,k)=1.
      if(type(mm).eq.cv)go to 35
      if(type(mm).eq.vv)go to 36
c  f7
      ans(it1,k)=-value(mm)
      go to 37
c current source type
33    if(kount.gt.nnode)go to 38
      f6(j,j)=1.
      if(type(mm).eq.vc)go to 36
      if(type(mm).eq.cc)go to 35
c  f2
      it2=ln+j
      ans(it1,it2)=-value(mm)
      go to 37
38    k=k+1
c  f7
      ans(it1,k)=1.
      if(type(mm).eq.vc)go to 36
      if(type(mm).eq.cc)go to 35
      f3(j,k)=-value(mm)
37    j=j+1
      if(kount.ne.icount(2))go to 32
```

```
      go to 39
c current controlled
35    if(itemp.gt.tp)go to 40
c  f5
      it2=nport+itemp
      ans(it1,it2)=-value(mm)
      go to 37
40    if(itemp.gt.nnode)go to 41
      it=itemp-tp
      f6(j,it)=-value(mm)
      go to 37
41    if(itemp.gt.icount(2))go to 42
      it=itemp-nnode
c  f7
      ans(it1,it)=-value(mm)
      go to 37
42    it=itemp-icount(2)
c  f8
      it2=nbr+tp+it
      ans(it1,it2)=-value(mm)
      go to 37
c voltage controlled
36    if(itemp.gt.tp)go to 43
c  f1
      it2=nbr+itemp
      ans(it1,it2)=-value(mm)
      go to 37
43    if(itemp.gt.nnode)go to 44
      it=itemp-tp
c  f2
      it2=ln+it
      ans(it1,it2)=-value(mm)
      go to 37
44    if(itemp.gt.icount(2))go to 45
      it=itemp-nnode
      f3(j,it)=-value(mm)
      go to 37
45    it=itemp-icount(2)
c  f4
      it2=nport+tp+it
      ans(it1,it2)=-value(mm)
      go to 37
39    if(debug.eq.0) go to 46
      if(ln.eq.0) go to 47
c  write f3 for debug run
      write(6,514)
      it1=1
48    it2=ln
      if((it2-it1).gt.10) go to 49
      if(it2.eq.it1) go to 47
      write(6,515)
      do 50 i=1,nport
50    write(6,516) (f3(i,j),j=it1,it2)
      go to 47
49    it2=it1+9
      write(6,515)
      do 51 i=1,nport
51    write(6,516) (f3(i,j),j=it1,it2)
      it1=it2+1
      go to 48
```

```
47    if(tp.eq.0) go to 52
c  write f6 for debug run
      write(6,517)
      it1=1
53    it2=tn
      if((it2-it1).gt.10) go to 54
      if(it2.eq.it1) go to 52
      write(6,515)
      do 55 i=1,nport
55    write(6,516) (f6(i,j),j=it1,it2)
      go to 52
54    it2=it1+9
      write(6,515)
      do 56 i=1,nport
56    write(6,516) (f6(i,j),j=it1,it2)
      it1=it2+1
      go to 53
52    write(6,518)
      call print(anscol,ansrow)
c
c zero out f6
c
46    if(tn.eq.0)go to 57
      do 58 j=1,tn
      kk=tp+j
      do 58 i=1,nport
      it1=port+i
      if(ln.eq.0) go to 59
c  change f7
      do 60 k=1,ln
      lk=nnode+k
60    ans(it1,k)=ans(it1,k)-(f6(i,j)*a(kk,lk))
59    if(lp.eq.0) go to 58
c  change f8
      do 61 k=1,lp
      lk=icount(2)+k
      it2=nbr+tp+k
61    ans(it1,it2)=ans(it1,it2)-(f6(i,j)*a(kk,lk))
58    continue
c
c zero out f3
c
57    if(ln.eq.0)go to 62
      do 63 j=1,ln
      lk=nnode+j
      do 63 i=1,nport
      it1=port+i
      if(tn.eq.0) go to 64
c  change f2
      do 65 k=1,tn
      kk=tp+k
      it2=ln+k
65    ans(it1,it2)=ans(it1,it2)-(f3(i,j)*(-a(kk,lk)))
64    if(tp.eq.0) go to 63
c  change f1
      do 66 k=1,tp
      it2=nbr+k
66    ans(it1,it2)=ans(it1,it2)-(f3(i,j)*(-a(k,lk)))
63    continue
c
```

```
c  fill ans matrix
c
62    if(debug.eq.0) go to 67
      write(6,519)
      call print(anscol,ansrow)
67    if(ln.eq.0.or.tp.eq.0)go to 68
c  store d1
      do 69 i=1,tp
      do 69 j=1,ln
      k=nnode+j
69    ans(i,j)=a(i,k)
68    lc=ln+1
      itemp=tp+1
      if(itemp.gt.port.or.lc.gt.nport)go to 70
c  store -d4 transpose
      do 71 i=itemp,port
      jj=lc+i-itemp+nnode
      do 71 j=lc,nport
      ii=j+1-lc+tp
71    ans(i,j)=-a(ii,jj)
70    if(tp.eq.0) go to 72
c  store unit matrix above f5
      do 73 i=1,tp
      ld=nport+i
73    ans(i,ld)=1.
72    if(lp.eq.0) go to 74
c  store unit matrix above f4
      ii=tp+1
      do 75 i=ii,port
      ld=nport+i
75    ans(i,ld)=1.
74    itemp=tp+1
      lf=ld+tp
      le=ld+1
      if(itemp.gt.port.or.le.gt.lf)go to 76
c  store -d2 transpose
      do 77 i=itemp,port
      jj=i-itemp+icount(2)+1
      do 77 j=le,lf
      ii=j+1-le
77    ans(i,j)=-a(ii,jj)
76    le=lf+lp
      ld=lf+1
      if(tp.eq.0.or.ld.gt.le)go to 78
c  store d2
      do 79 i=1,tp
      do 79 j=ld,le
      k=icount(2)+i+j-ld
79    ans(i,j)=a(i,k)
78    if(debug.eq.0) go to 80
      write(6,520)
      call print(anscol,ansrow)
c
c  reduce ans matrix to echelon form
c
80    call raech(nbr,anscol,anscol,1,1,zero)
      if(debug.eq.0) go to 81
      write(6,521)
      call print(anscol,ansrow)
81    do 82 i=1,nbr
```

```
      do 82 j=1,nport
      ii=nbr+1-i
      if(abs(ans(ii,j)).le.zero) ans(ii,j)=0.0
      if (ans(ii,j).ne.0.)go to 83
82    continue
83    ii=ii+1
c
c  fill column heading vector for final print out
c
      j=0
      if(tp.eq.0) go to 84
      do 85 i=1,tp
      it=2*i
      header(it)=br(i)
      header(it-1)=c
      i2=2*(port+i)
      header(i2)=br(i)
85    header(i2-1)=v
84    if(lp.eq.0) go to 86
      j=tp
      do 87 i=1,lp
      j=j+1
      k=i+icount(2)
      it=2*j
      header(it)=br(k)
      header(it-1)=v
      i2=2*(port+tp+i)
      header(i2)=br(k)
87    header(i2-1)=c
86    it=4*port
      nport1=nport+1
      do 88 i=ii,nbr
      do 88 j=nport1,anscol
88    if(abs(ans(i,j)).le.zero) ans(i,j)=0.0
      if(debug.eq.0) go to 89
c  print final ans matrix for debug run
      call prnt1(it,nport1,anscol,ii,nbr)
89    if (ii.eq.nbr) go to 90
c
c  back substitute final answer matrix
c
      it1=ansrow-ii+1
      it2=ii+1
      do 91 i=it2,ansrow
c  ans(irw,icl) is pivot element using to zero elements above
      irw=ansrow+it2-i
      icl=nport+it1+it2-i
      it3=irw-1
c  j=row zeroing out above pivot
      do 91 j=ii,it3
      b=ans(j,icl)
c  k=column changing of jth row
      do 91 k=icl,anscol
91    ans(j,k)=ans(j,k)-b*ans(irw,k)
90    do 92 i=ii,nbr
      do 92 j=nport1,anscol
92    if(abs(ans(i,j)).le.zero) ans(i,j)=0.0
c
c  print final ans matrix
c
```

```
      if(iprnt.eq.0) go to 499
      call prntl(it,nportl,anscol,ii,nbr)
499   return
500   format (80a1)
501   format(1h1/1x,80a1//////1x,'network description',///1x,'branch
     * from   to      element   element   control'/' number    node  node
     *     type      value    branch')
502   format(3i3,a2,f10.3,i3)
503   format(2x,i3,6x,i3,3x,i3,7x,a2,4x,e10.3,3x,i3)
504   format(1h0//,' zero = ',e10.3)
505   format(///' a matrix')
506   format(1h0,50i3)
507   format(1h0///)
508   format(1h0'tree port branches'/50(1x,i2))
509   format(1h0,'tree non-port branches'/50(1x,i2))
510   format(1h0,'link non-port branches'/50(1x,i2))
511   format(1h0,'link port branches'/50(1x,i2))
512   format(1h0,'tp = ',i3/' tn = ',i3/' ln = ',i3/' lp = ',i3)
513   format(1h0,'br',50(1x,i2))
514   format(///' f3 before zeroing')
515   format(1x)
516   format(1x,10(e11.4,1x))
517   format(///' f6 before zeroing')
518   format(///' ans matrix before zeroing')
519   format(///' ans matrix after zeroing')
520   format(///' ans matrix with d values filled in')
521   format(///' ans matrix reduced to echelon form')
c
      end
c---------------------------------------------------------------------
      subroutine ftree(nrow,ncol,indcol)
c
c subroutine ftree takes the matrix a, applies subroutine iaech and finds
c the independent columns in a closest to the left. these independent
c columns make up the tree branches.
c---------------------------------------------------------------------
c
      integer a,indcol(nrow),col,temp
      common a(40,90)
      l=1
      temp=1
      call iaech(nrow,ncol)
c step through rows
      do 1 k=1,nrow
c step through columns
      do 2 j=temp,ncol
c find independent columns
c test if element equal to one
      if (a(k,j).ne.1) go to 2
c record independent column number
      indcol(l)=j
      l=l+1
      temp=j+1
      go to 1
2     continue
1     continue
      return
      end
c
c---------------------------------------------------------------------
```

```fortran
      subroutine iaech(nrow,ncol)
c
c subroutine iaech manipulates matrix a into echelon form
c
c-----------------------------------------------------------------------
      integer a,c,g,gplus1,p,b
      common a(40,90)
      c=1
      g=1
2     do 1 i=g,nrow
      if(a(i,c).eq.0)go to 1
c interchange i and g row to get nonzero pivot
      if(i.eq.g) go to 3
      do 4 k=c,ncol
      b=a(i,k)
      a(i,k)=a(g,k)
      a(g,k)=b
4     continue
c normalize row to get positive number for pivot
3     if(a(g,c).gt.0) go to 5
      do 6 k=c,ncol
6     a(g,k)=-a(g,k)
5     if(g.ge.nrow) return
c zero column below pivot
      gplus1=g+1
      do 7 p=gplus1,nrow
      b=a(p,c)
      if(b.eq.0)go to 7
      do 8 k=c,ncol
8     a(p,k)=-b*a(g,k)+a(p,k)
7     continue
      g=g+1
      c=c+1
      go to 2
1     continue
      if(g.gt.nrow) return
      c=c+1
      go to 2
      end
c
c-----------------------------------------------------------------------
c
      subroutine raech(m,n,mark,row1,col1,zero)
      common ia(40,90),a(90,180)
      integer c,g,gplus1,p,row1,col1
c
c  raech performs row operations on a to reduce a to echelon form
c
c  columns col1 to mark are reduced to row echelon form while the row
c  operations are carried out on the rows from mark + 1 to n
c  rows row1 to m are reduced to row echelon form
c  g=row determining pivot point in
c  c=column determining pivot point in
c
c-----------------------------------------------------------------------
c
      c=col1-1
      g=row1
1     if(c.eq.mark) return
      c=c+1
```

```
c   find the max nonzero element in the c column below and including pivot
       i=0
       tmz=0.0
       do 2 j=g,m
       if(abs(a(j,c)).le.zero) a(j,c)=0.0
       if(abs(a(j,c)).le.tmz) go to 2
       tmz=abs(a(j,c))
       i=j
2      continue
       if(tmz.eq.0.0) go to 1
c   if the nonzero element is in the pivot row, do not exchange rows
       if(i.eq.g)go to 3
c   exchange pivot row with row having nonzero element in pivot column
       do 4 k=c,n
       b=a(i,k)
       a(i,k)=a(g,k)
4      a(g,k)=b
c   check if pivot point already normalized to 1
3      if(a(g,c).eq.1.)go to 5
c   normalize pivot row
       alpha=a(g,c)
       do 6 k=c,n
       a(g,k)=a(g,k)/alpha
6      if(abs(a(g,k)).le.zero) a(g,k)=0.0
c   check if just normalized pivot in last row
5      if(g.ge.m)return
c   zero the elements below the pivot
       gplus1=g+1
       do 7 p=gplus1,m
       b=a(p,c)
       if(abs(a(p,c)).le.zero) a(p,c)=0.0
       if(abs(a(p,c)).eq.0.0) go to 7
       do 8 k=c,n
8      a(p,k)=-b*a(g,k)+a(p,k)
7      continue
       if(g.ge.m) return
       g=g+1
       go to 1
       end
c
c-----------------------------------------------------------------
c
       subroutine print(anscol,ansrow)
c
c   subroutine print prints the entire ans matrix
c
c-----------------------------------------------------------------
c
c
c   prints ansrow rows by anscol columns
       integer a(40,90),anscol,ansrow
       common a,ans(90,190)
       it1=1
1      it2=anscol
       if((it2-it1).gt.9 ) go to 2
       if(it2.eq.it1) return
c   less than 10 columns left to print
       write(6,500)
       do 3 i=1,ansrow
3      write(6,501) (ans(i,j),j=it1,it2)
```

```
      return
2     it2=it1+9
c  more thrn 10 columns left to print
      write(6,500)
      do 4 i=1,ansrow
4     write(6,501) (ans(i,j),j=it1,it2)
      it1=it2+1
      go to 1
500   format(1x)
501   format(1x,10(e11.4,1x))
      end
c
c-----------------------------------------------------------------------
      subroutine prnt1(hdr,acl1,acl2,arw1,arw2)
c
c  subroutine prnt1 prints only the desired part of the ans matrix
c  describing the port equations along with the column headings
c
c-----------------------------------------------------------------------
c
      integer a(40,90),header(320),acl1,acl2,arw1,arw2,hdr
      common a,ans(90,180),header
      itm2=acl1-1
      it1=1
1     it2=hdr
      if((it2-it1).gt.19) go to 2
      if(it2.eq.it1) return
c  less or equal 10 columns to print
      write(6,500) (header(i),i=it1,it2)
      itm1=itm2+1
      do 3 i=arw1,arw2
3     write(6,501) (ans(i,j),j=itm1,acl2)
      return
2     it2=it1+19
c  more than 10 columns to print
      write(6,500)(header(i),i=it1,it2)
      itm1=itm2+1
      itm2=itm1+9
      do 4 i=arw1,arw2
4     write(6,501) (ans(i,j),j=itm1,itm2)
      it1=it2+1
      go to 1
500   format(1h0,10(4x,a1,i2,5x))
501   format(1h0,10(e11.4,1x))
      end
#eor
```

```
c-----------------------------------------------------------------------
c
c       Subroutine Lemke for solving the complementarity problem.
c       It utilizes the subroutines (matrix, initia, newbas, sort,
c       pivot and pprint.
c
c-----------------------------------------------------------------------
      subroutine lemke(n)
c
c
      common/004/am,q,w,z,l1,b,nl1,nl2,a,ne1,ne2,ir,mbasis,izr
      dimension am(50,50),q(50),b(50,50),a(50),mbasis(100)
      dimension w(50),z(50)
c description of parameters in common
c   am       a two dimensional array containing the
c            elements of matrix m.
c   q        a singly subscripted array containing the
c            elements of vector q.
c   l1       an integer variable indicating the number of
c            iterations taken for each problem.
c   b        a two dimensional array containing the
c            elements of the inverse of the current basis.
c   w        a singly subscripted array containing the values
c            of w variables in each solution.
c   z        a singly subscripted array containing the values
c            of z variables in each solution.
c   nl1      an integer variable taking value 1 or 2 depend-
c            ing on whether variable w or z leaves the basis
c   ne1      similar to nl1 but indicates variable entering
c   nl2      an integer variable indicating what component
c            of w or z variable leaves the basis.
c   ne2      similar to nl2 but indicates variable entering
c   a        a singly subscripted array containing the
c            elements of the transformed column that is
c            entering the basis.
c   ir       an integer variable denoting the pivot row at
c            each iteration. also used to indicate termina-
c            tion of a problem by giving it a value of 1000.
c   mbasis   a singly subscripted array-indicator for the
c            basic variables. two indicators are used for
c            each basic variable-one indicating whether
c            it is a w or z and another indicating what
c            component of w or z.
c   n        integer variable indicating problem size
c
c initialize basis inverse.
      do 9 j=1,n
      do 7 i=1,n
      if(i.eq.j)go to 8
      b(i,j)=0.0
      go to 7
8     b(i,j)=1.0
7     continue
9     continue
c parameter n indicates the problem size
      call initia(n)
c since for any problem termination can occur in initia,
c newbas or sort subroutine,the value of ir is matched with
c 1000 to check whether to continue or go to next problem.
      if(ir.eq.1000) return
```

```
   50 call newbas(n)
      if(ir.eq.1000)return
      call sort(n)
      if(ir.ge.1000)return
      call pivot(n)
      go to 50
      end
c
c-----------------------------------------------------------------------
      subroutine initia(n)
c purpose-to find the initial almost complementary solution
c          by adding an artificial variable zo.
c
c-----------------------------------------------------------------------
c
      common/004/am,q,w,z,l1,b,nl1,nl2,a,ne1,ne2,ir,mbasis,izr
      dimension am(50,50),q(50),b(50,50),a(50),mbasis(100)
      dimension w(50),z(50)
c set z0 equal to the most negative q(i)
      i=1
      j=2
    9 if(q(i) .le. q(j))go to 18
      i=j
   18 j=j+1
      if(j .le. n)go to 9
c update q vector
      ir=i
      t1=-q(ir)
      if(t1.le.0.0) go to 1000
      do 10 i=1,n
      q(i)=q(i)+t1
   10 continue
      q(ir)=t1
c update basis inverse and indicator vector
c of basic variables.
      do 12 j=1,n
      b(j,ir)=-1.0
      w(j)=q(j)
      z(j)=0.0
      mbasis(j)=1
      l=n+j
      mbasis(l)=j
   12 continue
      izr=ir
      nl1=1
      l=n+ir
      nl2=ir
      mbasis(ir)=3
      mbasis(l)=0
      w(ir)=0.0
      z0=q(ir)
      l1=1
      return
c
c
1000  ir=1000
      do 1010 i=1,n
      mbasis(i)=1
      mbasis(i+n)=i
1010  w(i)=q(i)
```

```
      return
      end
c
c------------------------------------------------------------------------
c
      subroutine newbas(n)
c
c purpose - to find the new basis column to enter in
c           terms of the current basis.
c------------------------------------------------------------------------
c
      common/004/am,q,w,z,l1,b,nl1,nl2,a,ne1,ne2,ir,mbasis
      dimension am(50,50),q(50),b(50,50),a(50),mbasis(100)
      dimension w(50),z(50)
c if nl1 is neither 1 nor 2 then the variable z0 leaves the
c basis indicating termination with a complementary solution
      if(nl1 .eq. 1)go to 20
      if(nl1 .eq.2)go to 21
c if the complementary solution and the number of iterations
c are to be printed set ipp=1 in the following statement
c
      ipp=0
      if(ipp.eq.1) then
      call pprint(n)
      endif
      ir=1000
      return
   20 ne1=2
      ne2=nl2
c update new basic column by multiplying by basis inverse.
      do 26 i=1,n
      t1=0.0
      do 28 j=1,n
   28 t1=t1-b(i,j)*am(j,ne2)
      a(i)=t1
   26 continue
      return
   21 ne1=1
      ne2=nl2
      do 29 i=1,n
      a(i)=b(i,ne2)
   29 continue
      return
      end
c
c------------------------------------------------------------------------
c
      subroutine sort(n)
c purpose - to find the pivot row for next iteration by the
c           use of (simplex-type) minimum ratic rule.
c
c------------------------------------------------------------------------
c
      common/004/am,q,w,z,l1,b,nl1,nl2,a,ne1,ne2,ir,mbasis,izr
      dimension am(50,50),q(50),b(50,50),a(50),mbasis(100)
      dimension w(50),z(50)
      amax=abs(a(1))
      do 10i=2,n
      if(amax.ge.abs(a(i)))go to 10
      amax=abs(a(i))
```

```
   10 continue
c set tol=amax*2.0**(-(b-11)) where b is the number of
c bits in the floating point mantissa as clasen suggests.
      tol=amax*2.0**(-27)
      i=1
   52 if(a(i).gt.tol)go to 51
      i=i+1
      if(i.gt.n)go to 9
      go to 52
   51 t1=q(i)/a(i)
      ir=i
   55 i=i+1
      if(i.gt.n)go to 56
      if(a(i).gt.tol)go to 54
      go to 55
   54 t2=q(i)/a(i)
      if(t2.ge.t1)go to 55
      ir=i
      t1=t2
      go to 55
   56 return
    9 if(q(izr).gt.tol)go to 57
      call pprint(n)
      ir=1000
      return
c failure of the ratio rule indicates termination with
c no complementary solution.
   57 print 250
  250 format(5x,37hproblem has no complementary solution)
      print 251,l1
  251 format(10x,13hiteration no.,i4)
      ir=1001
      return
      end
c
c-----------------------------------------------------------------------
c
      subroutine pivot(n)
c
c purpose - to perform the pivot operation by updating the
c              inverse of the basis and q vector.
c
c-----------------------------------------------------------------------
      common/004/am,q,w,z,l1,b,nl1,nl2,a,ne1,ne2,ir,mbasis
      dimension am(50,50),q(50),b(50,50),a(50),mbasis(100)
      dimension w(50),z(50)
      do 30 i=1,n
   30 b(ir,i)=b(ir,i)/a(ir)
      q(ir)=q(ir)/a(ir)
      do 31 i=1,n
      if(i.eq.ir)go to 31
      q(i)=q(i)-q(ir)*a(i)
      do 32 j=1,n
      b(i,j)=b(i,j)-b(ir,j)*a(i)
   32 continue
   31 continue
c update the indicator vector of basic variables
      nl1=mbasis(ir)
      l=n+ir
      nl2=mbasis(l)
```

```
      mbasis(ir)=ne1
      mbasis(l)=ne2
      l1=l1+1
      return
      end
c
c----------------------------------------------------------------------
c
      subroutine pprint(n)
c purpose - to print the current solution to complementary
c           problem and the iteration number.
c
c----------------------------------------------------------------------
c
      common/004/am,c,w,z,l1,b,nl1,nl2,a,ne1,ne2,ir,mbasis
      dimension am(50,50),q(50),b(50,50),a(50),mbasis(100)
      dimension w(50),z(50)
c zero the variable values.
      do 35 i=1,n
      w(i)=0.0
   35 z(i)=0.0
c
      i=n+1
      j=1
   42 k1=mbasis(i)
      k2=mbasis(j)
      if(q(j).ge.0.0)go to 45
      q(j)=0.0
   45 if(k2.eq.1)go to 40
      z(k1)=q(j)
      go to 41
   40 w(k1)=q(j)
   41 i=i+1
      j=j+1
      if(j.le.n)go to 42
      do 44 i=1,n
      write (6,601) i,w(i),z(i)
 601   format(1x,i1,2f14.4)
 44   continue
      write(6,602)l1
 602  format(1x,'number of pivots required', i3)
      return
      end
#eor
```

```
c-----------------------------------------------------------------------
c     subroutine ambset for quantizing the test node voltages into
c     ranges centered around the voltage value due to some fault
c     condition.
c
c     Input and internal variables
c
c     nvt      number of test nodes - input
c     nft      number of faults - input
c     nff      vector containing fault numbers as specified in the program
c              input
c     ch       array containing branch names (4 characters each)
c     n4       starting location of the test nodes in the array ch
c     amr      voltage range - input, default=1v
c     rng      array containing boundaries of voltage ranges
c     iout     array used in printing the one-zero form of the
c              ambiguity set-fault table.
c     nset     no of sets in the corresoonding test nodes.
c     ans      array containing the vt-fault table.
c     aa       integer working array
c     am       real working array
c
c-----------------------------------------------------------------------
      subroutine ambset(nvt,nft,ch,n4,amr,nff)
      dimension rng(2,20),iout(20,40),nset(40),icntr(20)
      dimension nff(30)
      common aa(40,90),ans(90,180)
      common/004/am(50,50)
      integer aa,ch(2,90)
c
      if(amr.eq.0.) amr=1.
      zero=10.**(-8.)
c clear flags and ambiguity set table (aa matrix)
      do 20 i=1,40
      do 20 j=1,40
      aa(i,j)=0
   20 continue
c set fault binary codes in aa column 40
      aa(1,40)=1
      do 5 j=2,nft
      aa(j,40)=aa(j-1,40)*2
    5 continue
c scan nodes in v-f table (ans matrix)
      do 200 n=1,nvt
c generate table of differences in am matrix
      do 10 j=1,nft
      k1=j+1
      do 9 k=k1,nft
      am(j,k)=ans(n,j)-ans(n,k)
    9 am(k,j)=am(j,k)
   10 continue
c clear output matrix
      do 25 i=1,20
      do 25 j=1,nft
   25 iout(i,j)=0
c set ambiguity set l=0 for node n
      l=0
c scan faults vertically
      do 100 j=1,nft
c if fault j has been scanned, skip and search
```

```
c for a new center to the next ambiguity set
      if(aa(j,39).ne.0)go to 100
c update flags and ambiguity set index
      aa(j,39)=1
      aa(j,38)=j
      l=l+1
      aa(j,37)=l
c set initial ambig set code and output matrix
      aa(l,n)=aa(j,40)
      iout(l,j)=1
c define ambig set range and center
      rng(1,l)=ans(n,j)-amr/2
      rng(2,l)=ans(n,j)+amr/2
      icntr(l)=j
c scan faults horizontally
      k1=j+1
c reset flags of updating overlapped ranges
      iflag1=0
      iflag2=0
      do 90 k=1,nft
c if fault k out of range skip it
      if(k.eq.j) go to 90
      if(abs(am(j,k)).ge.(amr/2)) go to 90
c if fault k has been scanned check if it
c is closer to the old or the new center
      ict=aa(k,38)
      ist=aa(k,37)
      if(aa(k,39).ne.1)go to 60
      w=abs(am(j,k))-abs(am(ict,k))
      if(w.ge.zero) go to 50
c if fault is closer to the new center add
c it to the new set and delete it from the
c old one then update flags and parameters
      aa(l,n)=aa(l,n).or.aa(k,40)
      aa(ist,n)=aa(ist,n)-aa(k,40)
      aa(k,38)=j
      aa(k,37)=l
      iout(l,k)=1
c update ambig set ranges
   50 if(am(ict,j).le.zero)go to 55
      if(iflag1.eq.1) go to 90
c set flag to acknowledge updating positive
c side overlap
      iflag1=1
      rng(2,l)=ans(n,j)+abs(am(j,ict))/2.
      rng(1,ist)=ans(n,ict)-abs(am(j,ict))/2.
      go to 90
   55 if(iflag2.eq.1)go to 90
      iflag2=1
      rng(1,l)=ans(n,j)-abs(am(ict,j))/2.
      rng(2,ist)=ans(n,ict)+abs(am(ict,j))/2.
      go to 90
c
c if fault has not been scanned and within range
c add it to the current set and update parameters
   60 aa(l,n)=aa(l,n).or.aa(k,40)
      iout(l,k)=1
      aa(k,39)=1
      aa(k,38)=j
      aa(k,37)=l
```

```
   90  continue
       if(l.eq.1) go to 100
       do 95 ll=1,l
       if((rng(1,l).lt.rng(2,ll)).and.(rng(1,l).gt.rng(1,ll))) then
       rng(1,l)=(rng(1,l)+rng(2,ll))/2.
       rng(2,ll)=rng(1,l)
       elseif((rng(2,l).gt.rng(1,ll)).and.(rng(2,l).lt.rng(2,ll))) then
       rng(2,l)=(rng(2,l)+rng(1,ll))/2.
       rng(1,ll)=rng(2,l)
       endif
   95  continue
  100  continue
       nset(n)=l
c write out centers, ranges and amiquity set codes
c for node n
       write(6,220) ch(1,n+n4),ch(2,n+n4)
       write(6,230)
       do 110 i=1,l
       m=icntr(i)
       icntr(i)=nff(m)
  110  write(6,240) i,icntr(i),(rng(j,i),j=1,2),(iout(i,j),j=1,nft)
c clear flags
       do 115 i=1,nft
       do 115 j=37,39
  115  aa(i,j)=0
  200  continue
  220  format(//2x,'node  ): ',2a2)
  230  format(//2x,'set',3x,'center',3x,'from',10x,'to',15x,'set code')
  240  format(/i4,i6,2(2x,e10.3),5x,40(i2))
       call ftcode(nvt,nft,nset,ch,n4,nff)
       return
       end
#eor
```

```
c-----------------------------------------------------------------------
c
c     subroutine ftcode for generating a numeric code fault dictiona-
c     ry based on the voltage ranges provided by the subroutine
c     "ambset".
c
c        Input and internal variables
c
c     nvt    number of test nodes - input
c     nft    number of faults - input
c     nset   number of ranges in the corresponding test node - input
c     ch     array containing branch names - input
c     n4     starting location of test nodes in ch - input
c     nff    vector containing fault numbers - input
c     icod   array containing the numeric codes on output
c     aa     integer working array
c
c----------------------------------------------------------------------
      subroutine ftcode(nvt,nft,nset,ch,n4,nff)
      dimension nset(40),iax(40),icod(40,40),itemp(40)
      integer ch(2,90)
      integer a,b,nff(30)
      common a(40,90)
c
c
c set node index to zero and find the node with maximum
c number of ambiguity sets
      idebug=0
      n=0
      lmax=nset(1)
      do 10 i=1,nvt
   10 lmax=max0(lmax,nset(i))
      lmax1=lmax
      if(idebug.eq.0) go to 201
      write(6,11) lmax1
   11 format(2x,'lmax1=',i20)
c move the corresponding set codes in 'a' to temporary
c storage and identify the node
  201 do 20 i=1,nvt
      if(nset(i).eq.lmax) go to 18
   20 continue
   18 i1=i
      nset(i1)=0
      do 19 j=1,lmax1
   19 itemp(j)=a(j,i1)
      n=n+1
      if(idebug.eq.0) go to 202
      write(6,41) n
      write(6,21) i1
   21 format(2x,'node',i5)
  202 nn=n
c initialize the fault code matrix icod(.,.)
c by the ambiguity set indeces of the first node
      do 22 i=1,lmax1
   22 icod(i,1)=i
      lf=lmax1
      if(nvt.gt.1) go to 30
      do 23 j=1,lmax1
   23 iax(j)=itemp(j)
      l='max1
```

```
      go to 100
c find the next maximum
   30 lmax=nset(1)
      do 35 i=1,nvt
   35 lmax=max0(lmax,nset(i))
      lmax2=lmax
      if(idebug.eq.0) go to 203
      write(6,36) lmax2
   36 format(2x,'lmax2=',i20)
  203 do 40 i=1,nvt
      if(nset(i).eq.lmax) go to 42
   40 continue
   42 nset(i)=0
      i2=i
      n=n+1
      nn=nn+1
      if(idebug.eq.0) go to 204
      write(6,41) n
      write(6,21) i2
   41 format(2x,'n=',i4)
c find the intersection of the am sets by logically
c anding the set codes
  204 l=1
      do 50 j=1,lmax1
c set a flag to detect first intersection
      ifl=0
      do 50 k=1,lmax2
      iax(l)=itemp(j).and.a(k,i2)
      if(iax(l).eq.0) go to 50
c otherwise update l and lf and insert the code
c icod(j,m),m=1,n-1 in the next locations (j+1)
c pushing own the rest of the codes up to the
c location lf. then add the set index k in the
c location icod(j,n) corresponding to set j -node n
      j1=lf-l+2
      n1=n-1
c skip pushing codes after first intersection only
      if(ifl.ne.0) go to 44
      icod(l,n)=k
      ifl=1
      go to 47
   44 do 45 li=1,j1
      do 45   m=1,n1
   45 icod(lf-li+2,m)=icod(lf-li+1,m)
      icod(l,n)=k
      lf=lf+1
   47 l=l+1
      if(l.le.nft) go to 50
      l=l-1
      go to 100
   50 continue
      l=l-1
      if(idebug.eq.0) go to 205
      do 48 ll=1,l
   48 write(6,49) (icod(ll,kk),kk=1,n)
   49 format(2x,'code',10i5)
c check if the new node is redundant
  205 if(l.ne.max0(lmax1,lmax2)) go to 70
c if the number of sets rsulting from inersction
c is not increased ignore the node and go to pick
```

```
c a new node
      n=n-1
      if(nn.eq.nvt) go to 70
      go to 30
c check if all faults have been isolated or all nodes
c are exhausted
   70 if(l.eq.nft.or.nn.eq.nvt) go to 100
c move result of intersection to temporary storage
      lmax1=l
      do 80 i=1,lmax1
   80 itemp(i)=iax(i)
      go to 30
  100 write(6,120)
      write(6,130) (ch(1,n4+m),ch(2,n4+m),m=1,n)
      do 95 i=1,l
      do 82 j=1,nft
      it1=iax(i).and.a(j,40)
      if(it1.eq.iax(i)) go to 81
      if(it1.eq.0) go to 82
      go to 83
   81 write(6,131) nff(j)
      write(6,140) (icod(i,m),m=1,n)
      go to 95
   82 continue
   83 m=0
      do 85 j=1,nft
      if(float(iax(i))/2.0.eq.float(iax(i)/2)) go to 84
      m=m+1
      a(k,41)=nff(j)
   84 iax(i)=iax(i)/2
   85 continue
      write(6,150) (a(k,41),k=1,m)
      write(6,140) (icod(i,j),j=1,n)
   95 continue
  120 format(//2x,'fault',23x,'fault code')
  130 format(/30x,20(2a2,1x))
  131 format(2x,'f',i2)
  140 format(30x,20(i3,2x))
  150 format(2x,'f',10(i2,','))
      stop
      end
#eor
```

**Appendix 3**

MIFTCOD: A Subroutine for Fault Isolation Under Multiple Test Input Conditions.

In case of applying different test input conditions to improve the level of isolation, there are two ways of handling the fault dictionary. The first way is to produce a fault code combining the input conditions, and that is to be used if the measurement data is available all at once. The other way is to generate a dictionary for every input. Then the isolation is achieved by correlating the dictionaries of the different inputs.

The subroutine MIFTCOD accepts the result of simulation for all input conditions after being classified into ambiguity ranges and produces the required fault codes for the individual inputs and the codes combining all inputs. This subroutine is not utilized by the program HAFDIC. Some extra effort has to be done in writing a small program that takes the results of HAFDIC simulation under different input conditions and then calls MIFICD to perform the required isolation. The program BASTEX shown next is an example of this.

**Example for Using MIFTCD:**

Table A3-1 contains the ambiguity sets of faults simulated in a circuit taken from [4], where two test inputs are used separately. To be able to supply the MIFTCD subroutine with the ambiguity sets in the required form, the ith fault is assigned the integer weight $2^i$ (see BASTEX program). This means that the integer word representing this fault will have the ith bit equal to one, while all other bits are zero. Adding or logical ORing of two faults will simply mean that there will be two bits equal to one in the integer word, and so on for more faults. The program output is shown next to the ambiguity sets table followed by the listing of MIFTCD.

```
c                               BASTEX
c
c        This program arranges the ambiguity sets of faults shown in
c        the accompanying table in a two dimensional array that provides
c        the required input to the subroutine MIFTCD.
c
      program bastex(input,output,tape5=input,tape6=output)
c this program generates the ambig sets of bastian7s paper
c example in the matrix a.
      dimension nset(4,20),ivt(20)
      integer a,c(20)
      common a(50,100)
      c(1)=1
      do 10 i=2,20
   10 c(i)=2*c(i-1)
      ics=c(1)
      do 5 i=2,20
    5 ics=ics+c(i)
      a(1,1)=c(3)+c(6)+c(7)+c(19)
      a(2,1)=ics-a(1,1)
      a(1,2)=c(3)
      a(2,2)=c(7)
      a(3,2)=c(15)
      a(4,2)=c(16)
      a(5,2)=ics-a(1,2)-a(2,2)-a(3,2)-a(4,2)
      a(1,3)=c(3)+c(6)+c(7)+c(15)
      a(2,3)=ics-a(1,3)
      a(1,4)=a(1,3)
      a(2,4)=c(19)
      a(3,4)=c(20)
      a(4,4)=ics-a(1,4)-a(2,4)-a(3,4)
      a(1,5)=a(1,3)+c(19)
      a(2,5)=ics-a(1,5)
      a(1,6)=a(1,5)
      a(2,6)=a(2,5)
      a(1,7)=a(1,5)
      a(2,7)=a(2,5)
      a(1,8)=a(1,3)+c(10)+c(12)+c(19)
      a(2,8)=ics-a(1,8)
      a(1,9)=c(2)
      a(2,9)=c(5)
      a(3,9)=c(13)
      a(4,9)=c(14)
      a(5,9)=ics-a(1,9)-a(2,9)-a(3,9)-a(4,9)
      a(1,10)=c(2)+c(4)+c(5)+c(13)
      a(2,10)=ics-a(1,10)
      a(1,11)=a(1,10)
      a(2,11)=c(9)
      a(3,11)=c(17)
      a(4,11)=c(18)
      a(5,11)=ics-a(1,11)-a(2,11)-a(3,11)-a(4,11)
      a(1,12)=a(1,10)+c(8)+c(9)+c(17)
      a(2,12)=ics-a(1,12)
      a(1,13)=a(1,12)
      a(2,13)=a(2,12)
      a(1,14)=a(1,12)
      a(2,14)=a(2,12)
      a(1,15)=a(1,12)
      a(2,15)=a(2,12)
      a(1,16)=a(1,12)
```

```
      a(2,16)=c(11)
      a(3,16)=ics-a(1,16)-a(2,16)
      ivt(1)=11
      ivt(2)=8
      ivt(3)=5
      ivt(4)=2
      ivt(5)=27
      ivt(6)=26
      ivt(7)=33
      ivt(8)=16
      nset(1,1)=2
      nset(1,2)=5
      nset(1,3)=2
      nset(1,4)=4
      nset(1,5)=2
      nset(1,6)=2
      nset(1,7)=2
      nset(1,8)=2
      nset(2,1)=5
      nset(2,2)=2
      nset(2,3)=5
      nset(2,4)=2
      nset(2,5)=2
      nset(2,6)=2
      nset(2,7)=2
      nset(2,8)=3
      nvt=8
      nft=20
      ni=2
      call miftcd(nvt,ivt,ni,nset,nft,c)
      stop
      end
#eor
```

| Node | Input | Set No. 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 11 | + 30 | 3,6,7,15,19 | nominal | | | |
| | −30 | 2 | 5 | 13 | 14 | nominal |
| 8 | + 30 | 3 | 7 | 15 | 16 | nominal |
| | −30 | 2,4,5,13 | nominal | | | |
| 5 | + 30 | 3,6,7,15 | nominal | | | |
| | −30 | 2,4,5,13 | 9 | 17 | 18 | nominal |
| 2 | + 30 | 3,6,7,15 | 19 | 20 | nominal | |
| | −30 | 2,4,5,8,9,13,17 | nominal | | | |
| 27 | + 30 | 3,6,7,15,19 | nominal | | | |
| | −30 | 2,4,5,8,9,13,17 | nominal | | | |
| 26 | + 30 | 3,6,7,15,19 | nominal | | | |
| | −30 | 2,4,5,8,9,13,17 | nominal | | | |
| 33 | + 30 | 3,6,7,15,19 | nominal | | | |
| | −30 | 2,4,5,8,9,15,17 | nominal | | | |
| 36 | + 30 | nominal | | | | |
| | −30 | nominal | | | | |
| 18 | + 30 | nominal | | | | |
| | −30 | nominal | | | | |
| 16 | + 30 | 3,6,7,10,12,15,19 | nominal | | | |
| | −30 | 2,4,5,8,9,15,17 | 11 | nominal | | |

combined inputs

---------------

| | v11,1v | v11,2v | 8,1v | 8,2v | 5,1v | 5,2v | 2,1v | 2,2v | v16,1v | v16,2v |
|---|---|---|---|---|---|---|---|---|---|---|
| f 3, | 1 | 5 | 1 | 2 | 1 | 5 | 1 | 2 | 1 | 3 |
| f 7, | 1 | 5 | 2 | 2 | 1 | 5 | 1 | 2 | 1 | 3 |
| f 6, | 1 | 5 | 5 | 2 | 1 | 5 | 1 | 2 | 1 | 3 |
| f19, | 1 | 5 | 5 | 2 | 2 | 5 | 2 | 2 | 1 | 3 |
| f 2, | 2 | 1 | 5 | 1 | 2 | 1 | 4 | 1 | 2 | 1 |
| f 5, | 2 | 2 | 5 | 1 | 2 | 1 | 4 | 1 | 2 | 1 |
| f13, | 2 | 3 | 5 | 1 | 2 | 1 | 4 | 1 | 2 | 1 |
| f14, | 2 | 4 | 5 | 2 | 2 | 5 | 4 | 2 | 2 | 3 |
| f15, | 2 | 5 | 3 | 2 | 1 | 5 | 1 | 2 | 1 | 3 |
| f16, | 2 | 5 | 4 | 2 | 2 | 5 | 4 | 2 | 2 | 3 |
| f 4, | 2 | 5 | 5 | 1 | 2 | 1 | 4 | 1 | 2 | 1 |
| f 9, | 2 | 5 | 5 | 2 | 2 | 2 | 4 | 1 | 2 | 1 |
| f17, | 2 | 5 | 5 | 2 | 2 | 3 | 4 | 1 | 2 | 1 |
| f18, | 2 | 5 | 5 | 2 | 2 | 4 | 4 | 2 | 2 | 3 |
| f20, | 2 | 5 | 5 | 2 | 2 | 5 | 3 | 2 | 2 | 3 |
| f 8, | 2 | 5 | 5 | 2 | 2 | 5 | 4 | 1 | 2 | 1 |
| f10,12, | 2 | 5 | 5 | 2 | 2 | 5 | 4 | 2 | 1 | 3 |
| f11, | 2 | 5 | 5 | 2 | 2 | 5 | 4 | 2 | 2 | 2 |
| f 1, | 2 | 5 | 5 | 2 | 2 | 5 | 4 | 2 | 2 | 3 |

separate input codes

input1

--------

fault code

| fault | v11 | v 8 | v 5 | v 2 | v16 | v |
|---|---|---|---|---|---|---|
| f 3, | 1 | 1 | 1 | 1 | 1 | |
| f 7, | 1 | 2 | 1 | 1 | 1 | |
| f 6, | 1 | 5 | 1 | 1 | 1 | |
| f19, | 1 | 5 | 2 | 2 | 1 | |

| | | | | | |
|---|---|---|---|---|---|
| f15, | 2 | 3 | 1 | 1 | 1 |
| f16, | 2 | 4 | 2 | 4 | 2 |
| f20, | 2 | 5 | 2 | 3 | 2 |
| f10,12, | 2 | 5 | 2 | 4 | 1 |
| f 1, 2, 4, 5, 8, 9,11,13,14,17, f18, | 2 | 5 | 2 | 4 | 2 |

input2

--------

fault code

| fault | v11 | v 8 | v 5 | v 2 | v16 | v |
|---|---|---|---|---|---|---|
| f 2, | 1 | 1 | 1 | 1 | 1 | |
| f 5, | 2 | 1 | 1 | 1 | 1 | |
| f13, | 3 | 1 | 1 | 1 | 1 | |
| f14, | 4 | 2 | 5 | 2 | 3 | |
| f 4, | 5 | 1 | 1 | 1 | 1 | |
| f 9, | 5 | 2 | 2 | 1 | 1 | |
| f17, | 5 | 2 | 3 | 1 | 1 | |
| f18, | 5 | 2 | 4 | 2 | 3 | |
| f 8, | 5 | 2 | 5 | 1 | 1 | |
| f11, | 5 | 2 | 5 | 2 | 2 | |
| f 1, 3, 6, 7,10,12,15,16,19,20, | 5 | 2 | 5 | 2 | 3 | |

```
c-----------------------------------------------------------------------
c
c       subroutine miftcd for generating a numeric code fault d
c       under multiple test input conditions.
c
c       input data
c
c       the ambiguity sets of faults have to be stored in a two
c       dimensional integer array such that every fault is represented
c       by a single bit in an integer word.  Every ambiguity set of
c       faults is then contained in one integer word.  Therefore the
c       number of faults is limited to the computer word length.
c       if the program utilizing miftcd is to be written in standard
c       fortran where bit manipulation is not available, the individual
c       bits can still be affected (see the example in appendix 3
c       fault diagnosis of nonlinear analog circuits vol5).
c       the arrangement of the ambiguity sets of nvt test nodes in
c       the array a should be as follows:
c       a(i,1)       ith ambig set of the first test node - input 1
c       a(i,2)       ith ambig set of the second test node - input 1
c       .....
c       a(i,nvt)     ith ambig set of the last test node - input 1
c       a(i,nvt+1)   ith ambig set of the first test node - input 2
c       a(i,nvt+2)   ith ambig set of the first test node - input 2
c       .....
c       a(i,2*nvt)   ith ambig set of the last test node -input 2
c       a(i,2*nvt+1)ith ambig set of the first test node - input 3
c
c       ......,etc.
c       input variables
c
c       nvt     number of test nodes
c       ivt     array containing the actual numbers of the test nodes
c               (for printing the fault dictionary)
c       ni      number of inputs
c       nset    array containing the numbers of ambig sets in the
c               corresponding test nodes
c       nft     number of faults
c        c      integer array containing integers (1,2,4,..2**nft)

c-----------------------------------------------------------------------
c
      subroutine miftcd(nvt,ivt,ni,nset,nft,c)
      dimension itemp(40),iax(40),ivt(nvt),icod(40,40)
      integer a,c(nft),cid(40,40)
      dimension nset(4,20),mset(20),node(20)
      common a(50,100)
c
c find the intersections of the ambiguity sets of every
c individual node ignoring null intersections
c
      nvt2=ni*nvt
c clear iax
      do 10 i=1,40
  10  iax(i)=0
      do 70 j=1,nvt
      j2=ni*(j-1)
      l=1
      n1=nset(1,j)
      n2=nset(2,j)
```

```
      do 20 i=1,n1
      do 20 k=1,n2
      iax(l)=a(i,j).anc.a(k,j+nvt)
      if(iax(l).eq.0) go to 20
      cid(l,j2+1)=i
      cid(l,j2+2)=k
      l=l+1
   20 continue
      l=l-1
c store the result (iax(i),i=1,l) in col #(nvt2+j) for
c further intersection with the sets of the next input
      do 30 i=1,l
   30 a(i,nvt2+j)=iax(i)
      mset(j)=l
c intersect a(i,nvt2+j) with next input a(k,j+nvt1)
c however if no of inputs .lt. 3 skip and go to next node
      if(ni.lt.3) go to 70
      do 60 n=3,ni
      n1=nset(n,j)
      nvt1=nvt*(n-1)
      lf=l
      m=1
      do 40 i=1,l
      ifl=0
      do 40 k=1,n1
      iax(m)=a(k,j+nvt1).and.a(i,j+nvt2)
      if(iax(m).eq.0) go to 40
      j1=lf-l+2
      nn1=n-1
      if(ifl.ne.0) go to 34
      cid(m,n+j2)=k
      ifl=1
      go to 37
   34 do 35 li=1,j1
      do 35 mi=1,nn1
   35 cid(lf-li+2,mi+j2)=cid(lf-li+1,mi+j2)
      cid(m,n+j2)=k
      lf=lf+1
   37 m=m+1
      if(m.gt.nft)go to 80
   40 continue
c adjust the resulting no of ambig sets
      l=m-1
c restore the resulting sets of the new intersection
      do 50 i=1,l
   50 a(i,j+nvt2)=iax(i)
   60 continue
c identify the no of combined inputs amb sets for node j
      mset(j)=l
   70 continue
      go to 100
c if current node isolates all faults acknowledge it
c and go to identify faults
   80 write(6,90) ivt(j)
   90 format(/2x,'node',i2,' is enough')
c
c output the combined input codes for the single sufficient node
c
     write(6,95) (ivt(j),n,n=1,ni)
   95 format(1x,'vt',i2,',',i1)
```

```
      do 96 i=1,nft
      do 96 k=1,nft
      it1=iax(i).and.c(k)
      if(it1.ne.iax(i)) go to 96
      write(6,340) (cid(i,m),m=1,ni),k
 96   continue
      n=1
      node(1)=j
      go to 200
c now the result of all inputs for every node j is available
c in a(i,nvt2+j) and the corresponding no of ambiguity sets
c is in mset(j). output mset(j). perform intersections of
c sets belonging to different nodes in a descending order of
c the no of ambig sets
 100  write(6,105) (mset(j),j=1,nvt)
 105  format(/2x,'combined input sets'/2x,20i3)
      lmax=mset(1)
      do 110 i=1,nvt
 110  lmax=max0(lmax,mset(i))
      lmax1=lmax
      lf=lmax1
c move the c...sponding sets to temporary storage and
c the node
      do 130 i=1,nvt
      if(mset(i).eq.lmax1) go to 135
 130  continue
 135  node(1)=i
      mset(i)=0
      n=1
      nn=1
      i1=i
      do 136 i=1,lmax1
 136  itemp(i)=a(i,i1+nvt2)
c load icod(.,.) with the combined input indeces of the first node
      j2=ni*(i1-1)
      do 138 i=1,lmax1
      do 138 m=1,ni
 138  icod(i,m)=cid(i,m+j2)
c find the next max and do the same
 139  lmax=mset(1)
      do 140 i=1,nvt
 140  lmax=max0(lmax,mset(i))
      lmax2=lmax
      if(nn.eq.nvt) go to 190
      if(lmax2.eq.0) stop
      do 150 i=1,nvt
      if(mset(i).eq.lmax2) go to 145
 150  continue
 145  mset(i)=0
      nn=nn+1
      n=n+1
      i2=i
      node(n)=i
      l=1
      j2=ni*(i2-1)
      jn=ni*(n-1)
      do 160 i=1,lmax1
      ifl=0
      do 160 k=1,lmax2
      iax(l)=itemp(i).and.a(k,i2+nvt2)
```

```
      if(iax(l).eq.0) go to 160
      j1=lf-l+2
      n1=n-1
      if(ifl.ne.0) go to 154
      do 153 m=1,ni
153   icod(l,m+jn)=cid(k,m+j2)
      ifl=1
      go to 157
154   do 155 li=1,j1
      do 155 mi=1,jn
155   icod(lf-li+2,mi)=icod(lf-li+1,mi)
      do 156 m=1,ni
156   icod(l,m+jn)=cid(k,m+j2)
      lf=lf+1
157   if(l.eq.nft) go to 190
      l=l+1
160   continue
      l=l-1
      write(6,161) l
161   format(2x,'l=',i5)
      if(l.ne.max0(lmax1,lmax2)) go to 170
c if current node is recundant ignore it and consider a new node
      n=n-1
      go to 139
c check if all nodes have been scanned
170   if(nn.eq.nvt) go to 190
c if yes go to find the fault codes otherwise move the sets
c to temporary storage and continue
      lmax1=l
      do 180 i=1,lmax1
180   itemp(i)=iax(i)
      go to 139
c write combined input fault table
190   write(6,191)
191   format(2x,'combined inputs')
      write(6,189)
189   format(1x,'---------------')
      do 192 i=1,n
      m=node(i)
192   mset(i)=ivt(m)
      write(6,193) ((mset(i),nm,nm=1,ni),i=1,n)
193   format(4x,30('v',i2,',',i1))
      nn=n*ni
c nn is the total no of entries in a row
      do 195 i=1,l
      do 182 j=1,nft
      it1=iax(i).and.c(j)
      if(it1.eq.iax(i)) go to 181
      if(it1.eq.0) go to 182
      go to 183
181   write(6,340) (icod(i,m),m=1,nn),j
      go to 195
182   continue
c find unisolated faults
183   m=0
      do 185 j=1,nft
      if(float(iax(i))/2.0.eq.float(iax(i)/2)) go to 184
      m=m+1
      a(m,100)=j
184   iax(i)=iax(i)/2
```

```
 185  continue
      write(6,340) (icoa(i,j),j=1,nn),(a(k,100),k=1,m)
 195  continue
c proceed to find the fault codes based on the selected set
c of nodes in node(j). the number of nodes = n. the no of
c ambig sets is already in nset(ni,j).
c the procedure is repeated for all inputs.
 200  write(6,201)
 201  format(/20x,'separate input codes')
      do 300 nii=1,ni
      nvt1=nvt*(nii-1)
      m=node(1)
      n1=nset(nii,m)
      lf=n1
      write(6,213) nvt1
 213  format(2x,'nvt1=',i5)
      write(6,214) ivt(m),n1
 214  format(2x,'vt',i3,' no of sets',i3)
c load icod with the indeces of the first node
c and move amb sets to temporary storage
      do 215 i=1,n1
 215  icod(i,1)=i
      do 220 i=1,n1
 220  itemp(i)=a(i,m+nvt1)
      if(n.ne.1) go to 225
c if one node is enough go to identify faults directly.
      do 222 i=1,n1
 222  iax(i)=itemp(i)
      go to 270
c start inersection
 225  do 260 nn=2,n
      l=1
      m=node(nn)
      n2=nset(nii,m)
      write(6,214) ivt(m),n2
      do 250 j=1,n1
c set flag to detect first intersection
      ifl=0
      do 250 k=1,n2
      iax(l)=itemp(j).and.a(k,m+nvt1)
      if(iax(l).eq.0) go to 250
c update indeces for stack pushing
      j1=lf-l+2
      nn1=nn-1
c skip pushing after the first intersection only
      if(ifl.ne.0) go to 244
      icod(l,nn)=k
      ifl=1
      go to 247
 244  do 245 li=1,j1
      do 245 mi=1,nn1
 245  icod(lf-li+2,mi)=icod(lf-li+1,mi)
      icod(l,nn)=k
      lf=lf+1
 247  l=l+1
 250  continue
c adjust resulting no of amb sets
      l=l-1
      wri.e(6,161) l
      do 251 i=1,l
```

```
  251   write(6,340) (icod(i,j),j=1,nn)
c move result of intersection to the temporary storage and
c continue to consider next node
        do 255 i=1,l
  255   itemp(i)=iax(i)
        n1=l
        lf=n1
  260   continue
c now iax(.) contains all sets representing isolated faults
c and icod contains corresponding fault codes. proceed to
c identify the isolated faults and print out the icod matrix
        write(6,370) nii
        write(6,380)
c write the actual node numbers
        do 275 i=1,n
        m=node(i)
  275   mset(i)=ivt(m)
  270   write(6,320)
        write(6,330) (mset(i),i=1,n)
c match sets against binary representation of the faults
        do 295 i=1,l
        do 282 j=1,nft
        it1=iax(i).and.c(j)
        if(it1.eq.iax(i)) go to 281
        if(it1.ec.0) go to 282
c if they do not match or anding not equal to zero then there
c must be a set of unisolated faults
        go to 283
  281   write(6,340) (icod(i,m),m=1,n),j
        go to 295
  282   continue
c find unisolated faults
  283   m=0
        do 285 j=1,nft
        if(float(iax(i))/2.0.eq.float(iax(i)/2)) go to 284
        m=m+1
        a(m,100)=j
c shift binary code once to the right
  284   iax(i)=iax(i)/2
  285   continue
        write(6,340) (icod(i,j),j=1,n),(a(k,100),k=1,m)
  295   continue
c continue to consider next input
  300   continue
  320   format(//2x,'fault code')
  330   format(/2x,20('vt',i2,1x),'fault')
  340   format(4x,20(i3,2x))
  370   format(//2x,'input',i1)
  380   format(/2x,'--------')
        return
        end
#eor
```

END

FILMED

5-83

DTIC